Universidad de las Ciencias Informáticas
FACULTAD 2

Título: *Técnicas para balancear la carga en Servidores Proxy, Aplicaciones Web y Base Datos.*

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Andro García Berdayes.

Tutor(es): Alberto Arce Martínez.

Co-tutor: Dionner Polanco.

Consultante: "[Insertar nombre consultante si existe]"

Asesor: "[Insertar nombre asesor si existe]"

26 de Junio de 2008

EN LA CIENCIA TODO EL CRÉDITO VA AL HOMBRE QUE CONVENCE AL MUNDO DE UNA IDEA, NO AL QUE LA CONCIBIÓ PRIMERO.

- William Osler

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

<Nombre autor>	**<nombre tutor>**
_____	_____
Firma del Autor	Firma del Tutor

Datos de Contacto

"[insertar breve curriculum e información de contacto del tutor]"
"[insertar breve curriculum e información de contacto del asesor]"
"[insertar breve curriculum e información de contacto del consultante]"

AGRADECIMIENTOS

Ante todo agradecerle a la revolución y a nuestro invicto Comandante en Jefe Fidel Castro por haber tenido la idea genial de reconstruir la base Lourdes en esta magnífica universidad.

Mis más grandes agradecimientos a mis padres y mi familia por todo el apoyo incondicional que me brindaron a lo largo de todos mis estudios.

Igualmente a mi tutor Alberto Arce por toda la ayuda que me brindo en la confección de la tesis en todo momento.

A mis amigos y amigas por aguantarme en el trayecto de estos cursos.

A los profesores que me mostraron el camino del saber y en general a todo aquel que de una forma u otra aporto su granito de arena.

Muchas Gracias

DEDICATORIA

Le dedico este trabajo a mis padres de manera bien especial, por ser los principales protagonistas de mi labor como estudiante y como hombre, por estar siempre en las buenas y malas, por ser las personas que más quiero, por cada consejo, cada opinión, cada discusión, por su incansable lucha de hacerme cada día una persona mejor.

RESUMEN

Se necesita balancear la carga en los servidores proxy, aplicaciones web y base datos en el nodo central, por lo que se propone una solución basada en el sistema operativo Red Hat que está instalado en estos servidores a través del Linux Virtual Server (LVS), es un software muy eficiente por las posibilidades que brinda de escalabilidad y disponibilidad en sus servicios y en el mundo de la internet muchos sitios web importantes balancea sus servidores con este software, utilizando la técnica de Direct-Routing entre las tres que existen por la ventaja de ser utilizada para red LAN y la cantidad de computadoras conectadas en red que tenemos en la UCI pues es muy eficiente, para enviar las peticiones de una manera organizada y más eficiente se escoge el algoritmo de Sheduling Weigwted Least-Connection (wlc), entre los diez más utilizados debido a que utiliza las ventajas de dos algoritmos, y no da oportunidad de que se formen cuellos de botellas en el servidor, también la herramienta Piranha que es la encargada de configurar el LVS, por su entorno web y por la facilidad de uso desde cualquier estación de trabajo, incluyendo una guía de los pasos a seguir para la instalación y configuración de la misma y de los componentes que necesita el LVS para su instalación y administración como es el caso del ipvsadm, entre otras opciones de monitorización de servidores y la persistencia.

PALABRAS CLAVES

Linux virtual server (LVS), escalabilidad, disponibilidad, balanceador, clúster, herramienta, técnicas, algoritmos, servidores, clientes, director, configuración, topología, arquitectura, paquetes, maquinas, software, piranha, monitorización, persistencia, encapsulamiento, datagrama.

Introducción

Debido al continuo crecimiento que ha tenido Internet en los últimos años hoy en día es difícil encontrarse con servidores que reciban miles o millones de conexiones diarias, teniendo que atender concurrentemente quizás a cientos o miles de ellas. Esto trae aparejada la necesidad de contar con una enorme cantidad de recursos computacionales para enfrentar las demandas crecientes por parte de los usuarios del servicio. Los requerimientos para los sistemas que hagan frente a estas demandas pueden resumirse en los siguientes **Escalabilidad**; cuando los requerimientos de un servicio aumentan, el sistema debe poder extenderse para poder atender todas las solicitudes sin que los tiempos de respuesta se vean afectados. **Alta disponibilidad**; el servicio debe estar disponible en todo momento, para lo cual debe ser tolerante a fallas. **Costos aceptables**; la inversión inicial y los costos de mantenimiento y expansión del sistema deben ser accesibles.

Para resolver estas dificultades se han desarrollado diversas soluciones basadas en el procesamiento paralelo. Estas se dividen en dos grandes enfoques: el fuertemente acoplado y, como contrapartida, el débilmente acoplado. El primer enfoque se basa en la utilización de equipos con múltiples procesadores que pueden ejecutar instrucciones concurrentemente y comparten el acceso a los datos locales, mientras que en el segundo se utilizan varios equipos independientes que colaboran entre sí para cumplir un objetivo. El enfoque de computación paralela fuertemente acoplada ha sido utilizado para todo tipo de tareas, sin embargo, esta solución, aunque efectiva y ampliamente aceptada, tiene serias desventajas principalmente en cuanto a escalabilidad y costos. En la actualidad están emergiendo numerosas opciones basadas en computación distribuida que se presentan como una alternativa razonable a los equipos de mediano y gran tamaño. Una de estas opciones es el Linux Virtual Server (LVS), un balanceador de carga, que permite implementar arquitecturas de disponibilidad y escalabilidad sobre sistemas GNU/Linux, independientemente de la plataforma hardware utilizada. Tiene por objetivo utilizar una red de computadoras u otros dispositivos de forma tal que sean accedidos desde el exterior como si fuera un único servidor, se conoce como servidor virtual. Esto permite asignar numerosos equipos de pequeño tamaño y costo reducido conectados mediante una red local de alto rendimiento para que presten el

servicio colaborando para lograr una capacidad computacional que crece casi linealmente con respecto a la capacidad computacional que agrega cada equipo en forma individual.[1]
La Universidad de las Ciencias Informáticas constituye hoy la segunda red más grande de nuestro país con más de 7200 PC conectadas a la red a través de la cual se sustentan los principales procesos de la misma como son la docencia, la producción y la investigación, además de brindar diferentes servicios a la comunidad universitaria que garantizan el funcionamiento de la institución. Debido a la cantidad de usuarios que se conecta diariamente para utilizar los servicios telemáticos, Internet, entre otros, se hace difícil a los servidores brindar un servicio estable y eficiente a todos los usuarios, por la cantidad de peticiones simultaneas que se realizan, lo cual conlleva a que se formen cuellos de botellas en la red, ocasionando lentitud en los diferentes servicios y en la transmisión de datos. La situación responde al siguiente Problema Científico: ¿Cómo eliminar los problemas de sobrecarga en los servidores que brindan servicios telemáticos para la UCI? El objeto de estudio analizado es el proceso de sobrecarga de los servidores. Seguidamente se analiza el Campo de acción que seria los servidores del nodo central de la Universidad de Ciencias Informáticas. El objetivo general que se persigue es: Diseñar una técnica para balancear la carga en los servidores del nodo central de la UCI con la utilización del Linux Virtual Server. Derivándose de este los siguientes objetivos específicos: Proponer una solución factible para los servidores. Realizar una documentación de la herramienta a utilizar.

Para organizar la investigación se platearon las siguientes preguntas científicas:

1. ¿Cuáles son los fundamentos teóricos y metodológicos que permiten la conceptualización del Linux Virtual Server (LVS)?

2. ¿Cuál es la técnica de balanceo de carga y el algoritmo a utilizar por LVS que permita el funcionamiento correcto en el nodo central?

3. ¿Cómo aplicar el Linux Virtual Server (LVS)?

Las tareas de investigación derivadas de este análisis fueron las siguientes:

1. Determinar los fundamentos teóricos y metodológicos que permiten la conceptualización del Linux Virtual Server (LVS).

2. Determinar la técnica de balanceo de carga y el algoritmo a utilizar por LVS que permita el funcionamiento correcto en el nodo central.

3. Elaborar una guía de cómo instalar y utilizar Linux Virtual Server (LVS).

Las características de la investigación condicionaron la selección de los métodos de investigación, entre los que se encuentran los del nivel teórico: análisis-síntesis, inducción-deducción y el enfoque de sistema.

Análisis síntesis: Este método ha estado presente en la revisión bibliográfica relativa al tema para obtener conocimientos sobre los diferentes aspectos de posible abordaje en el análisis de las mismas y elaborar el marco teórico conceptual sobre el objeto de estudio.

Inducción - deducción: Este método ha estado presente en los análisis realizados durante la revisión teórica metodológica, así como la propuesta de la estrategia donde diferentes elementos han permitido arribar a conclusiones. Entre los métodos empíricos figura: la observación y la entrevista.

La tesis está estructurada en tres capítulos donde se abordan los elementos del diseño teórico de la investigación, un primer capítulo que examina los aspectos teóricos y metodológicos sobre el Linux Virtual Server (LVS), un segundo capítulo que expone la técnica de balanceo de carga y el algoritmo a utilizar para un mejor funcionamiento de este sistema y el tercer capítulo muestra la guía o pasos a emplear para la instalación de este software.

Posibles resultados

Dándole un cumplimiento a los objetivos del trabajo se puede agregar que los posibles resultados son. Identificar la técnica de balanceo de carga y el algoritmo a utilizar por LVS que permita el funcionamiento correcto en el nodo central**.** Elaboración una guía para del Linux Virtual Server (LVS) en el nodo central, esto trae consigo mejorar los servicios telemáticos de

la UCI, también el tráfico de datos por la red se hace más eficiente. Las peticiones de los usuarios a los servicios son respondidas con más rapidez, se eliminan los cuellos de botellas, y Los usuarios en la UCI tendrían acceso más rápido a los datos que necesitan.

CAPITULO 1 FUNDAMENTOS TEORICOS

¿Cuáles son los fundamentos teóricos y metodológicos que permiten la conceptualización del Linux Virtual Server (LVS)?

Introducción

Linux Virtual Server (LVS) es utilizada por muchas compañías para balancear la carga de sus servidores. Es una técnica libre que brinda la posibilidad de instalación y mantenimiento con un conocimiento básico de Linux. Algunos países como España, México, Brasil, Ecuador, Argentina brindan información en sitios de cómo utilizar la herramienta sugiriendo el uso del software por los resultados óptimos que han obtenido.

1 Actualidad del LVS

El software de Linux Virtual Server se está utilizando en la actualidad en entornos de producción como las web del portal de Linux Linux.COM (http://www.linux.com), el portal de desarrollo SourceForge (http://www.sourcefoge.net), la web del integrador de servidores Linux VA Linux Systems, Inc. (http://www.valinux.com), la web de la empresa de vídeo y multimedia para la red Real Networks, Inc. (http://www.rela.com), o la web dedicada al análisis y comentarios de hardware AnandTech (http://anandtech.com) entre otros sitios de gran interés (Ver Anexo1.2). [3].Esta herramienta se ha comentado y discutido en foro de discusión con buenas opiniones por sus resultados.

Existen varios balanceadores de carga en el mundo como por ejemplo: El Juniper DX3680 no es un sistema que alguien compre solo para el balanceo de carga por su precio de $70 485 dólares. Hay balanceadores de carga mucho menos caros que crean clústeres para servidores Web, pero no tienen las sofisticadas funciones de motor de reglas y de aceleración, así como su habilidad para proporcionar autenticación para proteger servidores Web y servidores de aplicaciones, y para añadir transparentemente cifrado SSL a un sitio Web sin rehacer el código. Otro ejemplo es el D-LINK router balanceador de carga dos puertos WAN mas cuarto puertos LAN y el DI-LB604 con dos puertas WAN 10/100Mbps RJ-45, Balance de

carga por bytes, por paquetes y por sesión, Respaldo automático de conexión a Internet, Control de accesos por grupos, Calidad de Servicio QoS. Además podemos encontrar otros tipos de balanceadores de carga a nivel de hardware que también son muy efectivos y de muy alta calidad como por ejemplo: Panda GateDefender se adapta a las necesidades de las pequeñas, medianas y grandes empresas, ajustando su capacidad de análisis a las comunicaciones de la red. Además, su sistema de balanceo de carga completamente automático, permite repartir el volumen de trabajo entre las diferentes unidades adicionales. Como resultado, se incrementa la escalabilidad y el rendimiento de la protección del perímetro de la red contra virus, spam y contenidos no deseados pero tiene la misma desventaja que poseen los antes mencionado en relación al LVS y es que son propietarios, son sistemas costosos y con LVS pues se resuelve el mismo problema que con ellos.

1.1 LVS en Cuba

En cuba LVS poco a poco se ha ido introduciendo en varios lugares del país, debido al aumento de usuarios principalmente en las universidades y en empresas de gran tamaño, pero todavía no se ha difundido con un mayor auge, debido a que es sobre la plataforma Linux y no hemos emigrado a gran escala. Las universidades de Santiago de Cuba y la Universidad de las Villas están haciendo un estudio para la instalación. Estas son hasta el momento algunas provincias que tienen un conocimiento en el tema. Específicamente en Ciudad de la Habana, la Universidad de las Ciencias Informáticas (UCI) tiene instalado el LVS en teleformación. En la Universidad de Ciencias Informáticas se puede observar en los servidores de tele formación una solución factible pero solo para el sistema operativo Debian independientemente que el balanceo de carga a través de LVS tiene los mismo principios básicos, pues la soluciones varían en dependencia del sistema operativo que se tenga instalado en el servidor ya que las herramientas a utilizar son diferentes y especificas para cada sistema operativos de Linux.

1.2 Clasificación de balanceadores de carga

Los balanceadores de carga se pueden clasificar dentro de 4 grandes grupos: los balanceadores de carga para servidores (server load balancing), los balanceadores de carga para firewalls (firewall load balancing), los balanceadores de caches (cache load balancing) y los balanceadores de carga entre sistemas servidores remotos (global server load balancing)

• server load balancing: Reparten la carga entre un grupo de servidores aumentando la capacidad del sistema. Solucionan el problema de la caída de un servidor y aumentan la escalabilidad del sistema.

• firewall load balancing: Reparten la carga entre un grupo de firewalls. Aumenta la capacidad y escalabilidad del sistema.

• cache load balancing: Descargan de trabajo a los servidores web al redireccionar las peticiones a sistemas de cache.

• global server load balancing: Permiten redireccionar las peticiones de los clientes hacia el servidor o centro de datos más idóneo para estos, mejorando los tiempos de respuesta y la calidad del servicio. La idoneidad del servidor elegido puede realizarse en función de la distancia (un servidor por país o región), o en base a otros factores como el tráfico, calidad del enlace, etc. [2]

1.2.1 Balanceadores hardware

Los balanceadores a nivel de hardware son máquinas con un propósito específico y solo son útiles para el balanceo de carga.

Ventajas

- Potencia.
- Estabilidad.
- Escalabilidad.

Inconvenientes

- Precio (equipo, mantenimiento, técnicos) muy elevados.

1.2.2 Balanceadores software

Los balanceadores a nivel de software son servidores configurados para hacer balanceo.

Ventajas

- Precio.
- Una vez no sea necesario el balanceador o se requiera uno más potente se puede reciclar para otras tareas.
- Escalabilidad

Inconvenientes

- Mayor tiempo de mantenimiento.
- Menor potencia.

1.3 ¿Porqué usar LVS?

● Económico: Pertenece a la industria del software libre por lo que puede ser usado en cualquier momento sin costo de nada
● Escalable: permite agregar o quitar “nodos” en forma transparente.
● Seguro: “esconde” nuestros servidores reales.
● Aliviana el trabajo de nuestros servidores.

1.4 Balance de carga.

Normalmente, un servicio de red funciona de la siguiente manera: los clientes solicitantes del servicio dirigen sus peticiones a un servidor. Éste responde ofreciéndoles el servicio solicitado en función de la petición. Las limitaciones de este modelo son dos: por un lado la disponibilidad del servicio, que está ligada al correcto funcionamiento de un único sistema servidor, y por otro el límite en la capacidad para atender, procesar y servir peticiones del mismo. Para ampliar la capacidad del servicio, hay dos opciones: o bien reemplazar el servidor por otro de mayores prestaciones, o bien añadir más servidores. En caso de optar por la segunda opción y dado que este cambio ha de ser transparente a los clientes, será necesario seguir ofreciendo el servicio sin variar el escenario anterior, es decir, manteniendo

la dirección IP a la que los clientes dirigían sus peticiones (IP Virtual o VIP). A este tipo de arquitectura se la conoce con el nombre de servidor virtual, cuyo dispositivo clave para ser implementada con éxito es el balanceador de carga. Nótese que, mediante un clúster de alta disponibilidad activo/pasivo también se consigue un servidor virtual, si bien este no ofrece un sistema escalable como el ofrecido por un balanceador de carga. [2]

1.4.1 Definición

El balanceador de carga es el dispositivo de red que se ubica entre los clientes y los servidores (figura 1), centralizando la recepción de peticiones. El balanceador de carga se limitará a reenviar las peticiones a los servidores reales, que son los encargados de procesarlas, incrementando de esta manera la capacidad de procesado de peticiones que tenía el antiguo sistema, basado es un solo servidor. El balanceador de carga toma las decisiones de reenvío de peticiones en función de un algoritmo de scheduling determinado. El método de reenvío puede implementarse de varias formas, en función de la arquitectura de red disponible para comunicar a los servidores reales con el balanceador de carga. (Ver Anexo 1).

1.4.2 Mejoras que aporta el balanceador de carga

El hecho de distribuir el trabajo entre diferentes máquinas permite que la capacidad de carga del servidor virtual sea muy superior a la que tendría un solo servidor real. Además, el balanceador de carga aporta escalabilidad, disponibilidad, facilidad de mantenimiento, seguridad y calidad de servicio.

1.4.2.1 Escalabilidad

Si la demanda de un servicio es muy elevada, se puede incrementar el número de servidores reales que prestan dicho servicio, de esta manera, con sólo aumentar el número de servidores reales, se aumenta el número de conexiones que puede atender el servidor virtual. El balanceador de carga distribuye las peticiones de los clientes entre todos los servidores reales disponibles, usando para ello algoritmos de distribución de la carga, que permiten aumentar la capacidad de proceso que tendría un solo servidor. Suponiendo un balanceo perfecto, la

capacidad del servidor virtual equivaldría a la suma de las capacidades de cada uno de los servidores reales por separado, pero en términos reales la capacidad del servidor virtual ronda entre el 80% y el 90%. [2]

1.4.2.2 Disponibilidad

La máquina encargada de balancear la carga hace un chequeo continuo de los servidores reales así como de las aplicaciones que corren en ellos. En caso de que un servidor real o una aplicación dejen de responder, el balanceador ya no le reenviará más peticiones hasta que el servicio esté disponible de nuevo. De esta forma, de cara al usuario final (cliente), el servicio ofrecido por el servidor virtual estará siempre disponible, incluso en el caso de que alguno de los servidores reales deje de prestar servicio. Si el sistema no dispone de los mecanismos necesarios, las conexiones que en ese momento tenía el servidor real fallido se perderán. En este sentido, el balanceador de carga hace automáticamente y con transparencia lo mismo que se haría manualmente, mediante configuraciones estáticas del balanceador de carga, pero disminuyendo notablemente los tiempos de respuesta ya que no se requiere de intervención humana para reconfigurar el balanceador. [2]

1.4.2.3 Gestión y flexibilidad en el mantenimiento

Si se necesita apagar algún servidor real o detener un servicio, debido a un cambio de hardware o a una actualización del software, el uso de un balanceador de carga permite hacerlo de forma controlada, en el sentido de que se dejan de reenviar las peticiones de los clientes a dicho servidor. De esta forma, es posible proceder a realizar las tareas de mantenimiento sin afectar a la disponibilidad del servicio. El proceso es totalmente transparente para los clientes, ya que siempre encontrarán disponibilidad del servicio en la IP habitual (VIP).

El balanceador de carga aporta flexibilidad en la gestión del servidor virtual porque posibilita cambiar los servicios que está ofreciendo cada uno de los servidores reales en función de las necesidades de cada momento.

Los balanceadores de carga permiten redireccionar las peticiones a los servidores reales con independencia del sistema operativo que éstos últimos ejecuten. Este hecho permite mezclar diferentes plataformas en un mismo servidor virtual. [2]

1.4.2.4 Seguridad

Los balanceadores de carga se encuentran entre los clientes y los servidores reales, pudiendo actuar como filtro ante posibles ataques. Adicionalmente, el uso de NAT incrementa la seguridad debido al hecho de que las direcciones IP privadas de los servidores reales no son alcanzables directamente desde Internet. Los clientes únicamente ven la VIP que proporciona el balanceador, en cuyo interior se aplicarán las reglas de filtrado. [2]

1.5 Linux Virtual Server.

1.5.1 ¿Qué es LVS?

Linux Virtual Server (LVS) es un clúster de máquinas que se muestran como un único servidor a los clientes. Este aparentemente único servidor es llamado “servidor virtual”. A los servidores integrantes del clúster se los conoce como real servers o servidores reales, y están bajo el control del balanceador de carga conocido como director. LVS se implementa mediante el parche para el kernel de Linux IP Virtual Server (IPVS).

.Esta permite asignar uno o más equipos como redirectores o balanceadores de carga para una red conformada por varios equipos. En el caso de los redirectores son también conocidos como servidores virtuales, ya que son percibidos por los clientes como servidores de una aplicación específica, mientras que los últimos son llamados servidores reales, debido a que en estos es donde se ejecuta realmente la aplicación servidora.[4]

¿Quién lo creo?

Es un proyecto de código abierto iniciado por Wensong Zhang en mayo de 1998. El objetivo es desarrollar un servidor Linux de alto rendimiento que proporcione buena escalabilidad, confiabilidad y robustez usando tecnología clustering. Actualmente, la labor principal del proyecto LVS es desarrollar un sistema IP avanzado de balanceo de carga por software (IPVS), balanceo de carga por software a nivel de aplicación y componentes para la gestión de clústeres. [4]

Otra definición

LVS es un avanzado balanceador de carga que permite la creación de clústeres de máquinas de una forma rápida y eficaz sin necesidad de hacer grandes inversiones en hardware dedicado. LVS es la solución ideal para ISPS y empresas que quieren ahorrar costes permitiendo tener tras una única IP pública muchos servidores reales y servicios de forma transparente de cara a los usuarios, además también se conocen varios conceptos pero todos con el mismo propósito como por ejemplo.(Ver Anexo 1.1). El principal componente del software es un conjunto de parches, es decir modificaciones, que se aplican al código fuente del núcleo del sistema operativo Linux y que le agregan nuevas funcionalidades. Para ello, este nuevo código interactúa con las rutinas de manejo del protocolo TCP/IP. En las distribuciones más nuevas de Linux es probable que los parches ya hayan sido aplicados al kernel estándar y se incluyan también las herramientas de usuario por lo que no se requerirá la instalación de software adicional a menos que se necesite una versión más reciente que incluya alguna característica novedosa o la corrección de un error que afecte en forma notoria el funcionamiento de la aplicación a utilizarse. En caso de compilarse el kernel de Linux, se requieren en forma genérica que las siguientes opciones se encuentren activadas –para la compilación estática o en forma de módulos. Se utiliza una arquitectura débilmente acoplada que permite utilizar un equipo como redirector de las conexiones entrantes al sistema distribuyéndolas de acuerdo a un algoritmo de scheduling determinado en un grupo de servidores. Esto permite que los distintos equipos compartan la carga de trabajo, pudiéndose

escalar el sistema simplemente agregando nuevos servidores a la red.

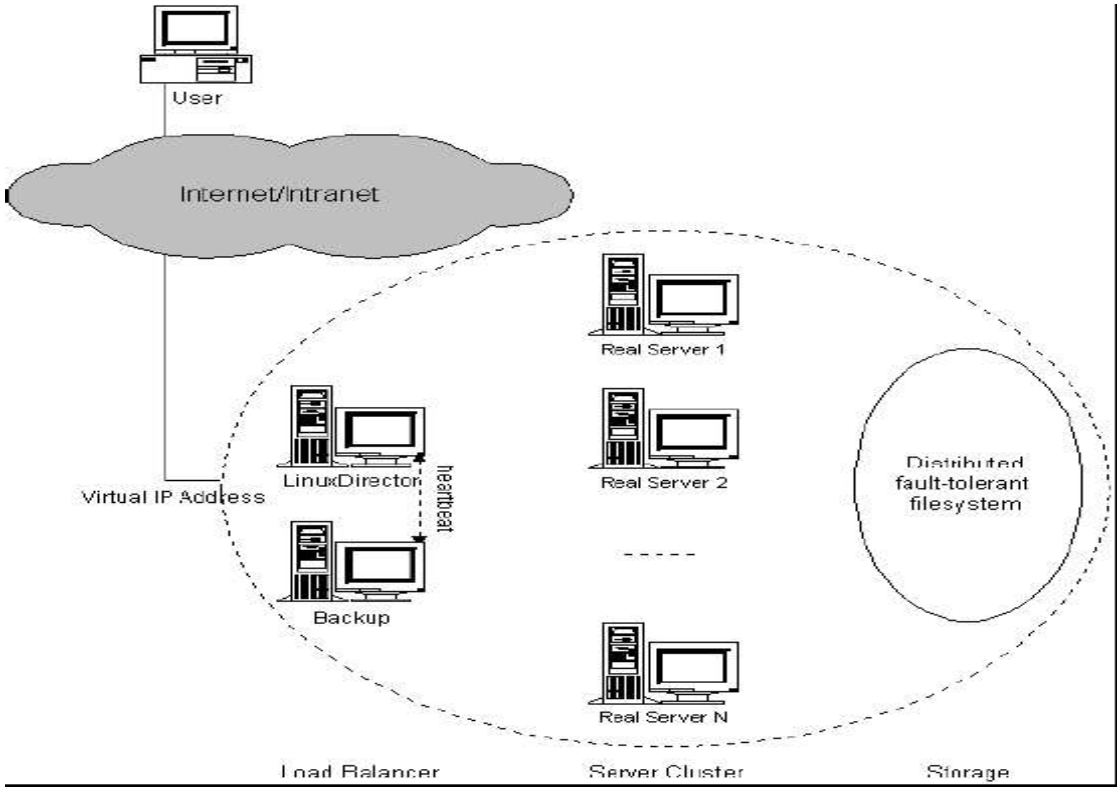


Figura 1 Linux Virtual Server

Como se ve en la figura 1 el sistema utiliza una arquitectura de tres capas formadas por el/los redirectores, los servidores reales y el sistema de archivos compartido (aunque este último componente no es estrictamente indispensable).El redirector es visto desde el exterior como el servidor ya que tiene asignada la dirección IP con que se publica el servicio. Puede recibir múltiples conexiones y redirigirlas a numerosos servidores reales según el o los criterios de balanceo de carga que se hayan adoptado. Dado que la aplicación servidora no se ejecuta en este equipo y que el software encargado de redireccionar las conexiones se ejecuta en su kernel disminuyendo drásticamente el overhead de procesamiento, es capaz de atender muchas más conexiones que las que podría atender si las conexiones se hicieran directamente a este equipo, manteniendo los tiempos de respuesta. [1]

1.5.2 Los servidores reales

Es un conjunto de servidores en cada uno de los cuales se ejecuta la aplicación. Esta puede ser cualquiera que utilice el protocolo TCP/IP. Sin embargo en el caso de que el servidor tenga que iniciar la apertura activa de conexiones con el cliente -por ejemplo FTP-, esto puede causar problemas, lo cual se soluciona a través de la incorporación de módulos específicos en el redirector para el manejo de la aplicación.

1.5.3 El repositorio de datos compartido

Aunque este no es un componente fundamental del sistema, provee la facilidad de poder administrar los datos en forma centralizada. Este repositorio común de datos podría no estar presente en el sistema, utilizándose un acceso local a los datos por parte de los servidores reales, pero de esta manera se haría muy dificultoso el proceso de modificar los datos de la aplicación como son por ejemplo los archivos HTML en un servidor Web. Debido a que estos repositorios serán accedidos por todos los servidores reales al mismo tiempo, es necesario que sean también escalables y tolerantes a fallas. Otro aspecto a tener en cuenta al implementar el sistema de archivos compartido es que, en caso de que la aplicación realizara cambios en los datos, es necesario un esquema de manejo de bloqueos, ya que las aplicaciones que se ejecutan en los servidores reales accederán a los datos en forma concurrente y es fundamental para la coherencia de los datos que no se permita a dos aplicaciones modificar un mismo dato compartido al mismo tiempo. Este esquema puede estar incorporado en el sistema de archivos compartido o puede ser manejado por alguna aplicación específica como por ejemplo un servidor de base de datos. [1]

1.6 Software libre

El modelo en el que se basa la industria del software de hoy día, difiere notablemente del marco con el que funcionan el resto de industrias “tradicionales”. Las licencias actuales de software, son en gran medida las responsables de esta situación, pues en el marco actual de soluciones software propietarias, las regulaciones sólo afectan al usuario, que está obligado a cumplir lo establecido en la licencia del software que utiliza, quedando la empresa

desarrolladora exenta de cualquier responsabilidad. En la industria tradicional es imposible pensar en una situación de esas características. Por ejemplo, cuando una empresa fabricante de coches vende un modelo al usuario, la empresa ha de fabricar el mismo en base a una estricta normativa que asegure la calidad y seguridad del producto. El cliente se compromete a utilizar el producto en base a lo que dictan las normas de circulación y la empresa, por otro lado, ofrece unas garantías si el cliente hace un uso normal del vehículo. En cambio, una licencia de software sólo implica compromiso por parte del usuario: no utilizar el software en más de una máquina o en máquinas diferentes con solo una licencia etc. Esto da lugar a situaciones poco razonables en las que un usuario debería pagar dos licencias para instalar un mismo programa, dos veces, en máquinas de su propiedad. El modelo libre establece sus propios mecanismos naturales para evitar situaciones monopolistas. El software libre no se ajusta al modelo actual de industria de software, ni en el modelo de ingeniería de software, utilizado para su desarrollo, ni en el tipo de las licencias utilizadas para su distribución. Del modelo de desarrollo utilizado se sabe muy poco, básicamente que es está distribuido globalmente y que el resultado funciona, en tanto que es un hecho que algunos programas libres funcionan igual o mejor que soluciones propietarias. Actualmente existen dos grandes corrientes en el mundo del software libre. Por un lado la liderada por Richard M. Stallman, creador de la Free Software Fundation (FSF), y por el otro la promovida por el movimiento Open Source. La primera basa sus motivaciones en consideraciones éticas, la segunda en motivos prácticos. En cualquier caso las dos confluyen el hecho de utilizar el mismo software. El software libre establece un modelo de costes diferentes al tradicional. Si la empresa utiliza el modelo propietario se le va todo el dinero en licencias, por el contrario, si la empresa utiliza el modelo libre, se gastará el dinero en personal o en empresas de soporte locales. El software libre surgió envuelto en una filosofía de derechos y de libertad que pretendía elevar el concepto de la creación de software a los niveles más altos de las virtudes humanas como son el altruismo y la cooperación. [2]

1.6.1 ¿Qué es software libre?

Software libre es aquel software que cumple las siguientes cuatro características:
Libertad de uso

Los programas distribuidos bajo una licencia libre, permiten al usuario utilizar en la forma que considere más oportuna y dándole plena libertad para que lo desee instalar, y hacerlo en tantos sistemas como quiera.

Libertad de redistribución

El autor permite la libre distribución del programa conjuntamente con la de su código fuente y su licencia. Copiar software libre no es ilegal, sino que además, es deseable y contribuye a su divulgación.

Libertad para la modificación del código

Cualquier usuario tiene plena libertad para modificar el código fuente del programa. Esto, trasladado a un entorno empresarial permite adaptar el programa a las necesidades especificas de la empresa, en cambio, si la empresa opta por una solución propietaria, y esta no se ajusta del todo a sus necesidades, deberá esperar a que la empresa suministradora de ese software decida implementar las características nuevas que la empresa requiere, si es que esta lo hace. Por otro lado, eso permite que cualquier empresa pueda dar soporte a un programa libre, si dispone de programadores preparados para ello, pese a no ser la creadora del programa. Así, dado que existen diferentes empresas que den soporte a un determinado programa, un empresario podría, eventualmente, cambiar de empresa si no está satisfecho con el soporte obtenido. En el escenario actual, en el cual las empresas suelen disponer de cuotas de mercado del 80% ó 90% para un determinado producto, esto no es posible ya que se depende de la empresa creadora para el soporte a su producto.

Libertad para redistribuir las modificaciones

Siempre y cuando las modificaciones se hagan también bajo una licencia libre, cualquiera puede seleccionar un puñado de software y redistribuirlo a aquellas empresas capaces de añadir sustanciales mejoras obteniendo mayor cuota de mercado, ya que software libre no implica que no se pueda cobrar por su distribución. Es entonces cuando aparece la competencia entre distribuciones y distribuidores que redundan en beneficio del usuario.

El término Software no Libre se emplea para referirse al software distribuido bajo una licencia de software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del “copyright”; el software dispuesto bajo una licencia de software libre

rescinde específicamente la mayoría de estos derechos reservados. (Stallman) La definición de Software Libre no contempla el asunto del precio; un eslogan frecuentemente usado es "libre como en libertad, no como en cerveza gratis" o en inglés "Free as in freedom, not as in free beer" (aludiendo a la ambigüedad del término inglés "free"), y es habitual ver a la venta CD de Software Libre como distribuciones Linux. Sin embargo, en esta situación, el comprador del CD tiene el derecho de copiarlo y redistribuirlo. El software gratis puede incluir restricciones que no se adaptan a la definición de Software Libre, por ejemplo, puede no incluir el código fuente, puede prohibir explícitamente a los distribuidores recibir una compensación a cambio. (Raymond) Para evitar la confusión, alguna gente utiliza los términos "libre" (Libre Software) y "gratis" (Gratis Software) para evitar la ambigüedad de la palabra inglesa "Free". (Libre, 2002) Sin embargo, estos términos alternativos son usados únicamente dentro del movimiento del software libre, aunque están extendiéndose lentamente hacia el resto del mundo. Otros defienden el uso del término Open Source Software (OSS, Software de Código Abierto, también llamado de Fuentes Abiertas). La principal diferencia entre los términos “Open Source” y “Free Software" es que éste último tiene en cuenta los aspectos éticos y filosóficos de la libertad, mientras que el "Open Source" se basa únicamente en los aspectos técnicos. (Junco, 2008).

Concluciones

En el capitulo se da respuesta a la pregunta inicial, se abordan los conceptos principales del LVS, se da un estado del arte del software. Tambien se apuntan temas de distintos balanceadores de carga propietarios con sus caracteristicas y costos que trae consigo una inclinacion a utilizar software libre por lo que se conceptualiza este tema, por el gran desarrollo social que brinda sin tener que gastar grandes sumas de dinero y resolver asi las mismas necesidades.

CAPITULO 2 TECNICAS Y ALGORITMOS

¿Cuál es la técnica de balanceo de carga y el algoritmo a utilizar por LVS que permita el funcionamiento correcto en el nodo central?

Introducción

LVS soporta tres técnicas o modos de trabajo distintos. Estos modos definen el método en que el balanceador retransmitirá la comunicación proveniente de los clientes a los servidores reales definidos. Además que presenta algoritmos de planificación para su funcionamiento interno. Tras este término se encuentra la manera en que LVS repartirá la carga de trabajo entre los distintos servidores reales. Actualmente LVS soporta 10 algoritmos distintos (más alguno especial). Definiremos sin adentrarnos en detalles los tres métodos en los siguientes párrafos de los cuales seleccionaremos 1 para nuestra propuesta y a la vez seleccionaremos los algoritmos de trabajo.

Como se mencionó anteriormente, Linux Virtual Server permite la implementación de un servicio virtual a través del uso de un balanceador de carga para una red formada por los servidores reales que ejecutan la aplicación. En esta arquitectura las peticiones de servicio de los clientes son enviados hacia el balanceador de carga. El primer paquete que le llega a este es uno de inicio de conexión, el cual tiene como destino la dirección IP que el redirector tiene asignada para el servicio virtual. El redirector examina la dirección IP y el puerto de destino, si coincide con alguno de los servicios publicados según la tabla de servicios virtuales elige un servidor real para ese servicio mediante el algoritmo de scheduling que haya sido seleccionado para ese servicio. Una vez que el servidor real ha sido seleccionado, se agrega la conexión a la tabla correspondiente y se reenvía el paquete al servidor real correspondiente haciendo uso de alguna de las tres técnicas de balanceo de carga que se describen en las siguientes secciones. La técnica a utilizar dependerá del servidor real elegido, ya que este parámetro se puede configurar independientemente para cada servidor real. Los paquetes que lleguen posteriormente y que pertenezcan a esta conexión serán identificados mediante la tabla de conexiones y luego reenviados al servidor real correspondiente. Al recibir los

paquetes el servidor real, este los percibe como provenientes del cliente, los procesa y luego devuelve la contestación al cliente. La forma en que es enviada esta información al cliente varía según la técnica de balanceo de carga utilizada, pero la aplicación percibirá siempre que está enviando los datos directamente al cliente. Estos demonios envían mensajes que pueden ser tipo ping o bien de prueba del servicio a intervalos regulares y si no reciben respuesta pueden proceder a eliminar un servidor real de la lista que se encuentra en el redirector o pueden iniciar la toma de posesión de la dirección IP del balanceador de carga para tomar su lugar. [1]

2.1 lvs mediante NAT

Este método NAT (Network Address Translation - Traducción de Direcciones de Red) para mapear servicios desde la IP del balanceador hacia las IPs internas y puertos de servicio de los servidores reales. Una petición realizada desde un cliente llegará al balanceador, este comprobará la dirección ip y puerto de destino del paquete y si estos coinciden con las tablas de servidores reales definidas en LVS modificará el paquete y lo inyectará de nuevo en la para que este llegue a su destino "real". Así mismo los paquetes de retorno de los servidores reales deben ser modificados y reinyectados para llegar al cliente original. Lógicamente esto sucede porque los servidores reales utilizan el balanceador LVS como Gateway de cara a Internet/Intranet.

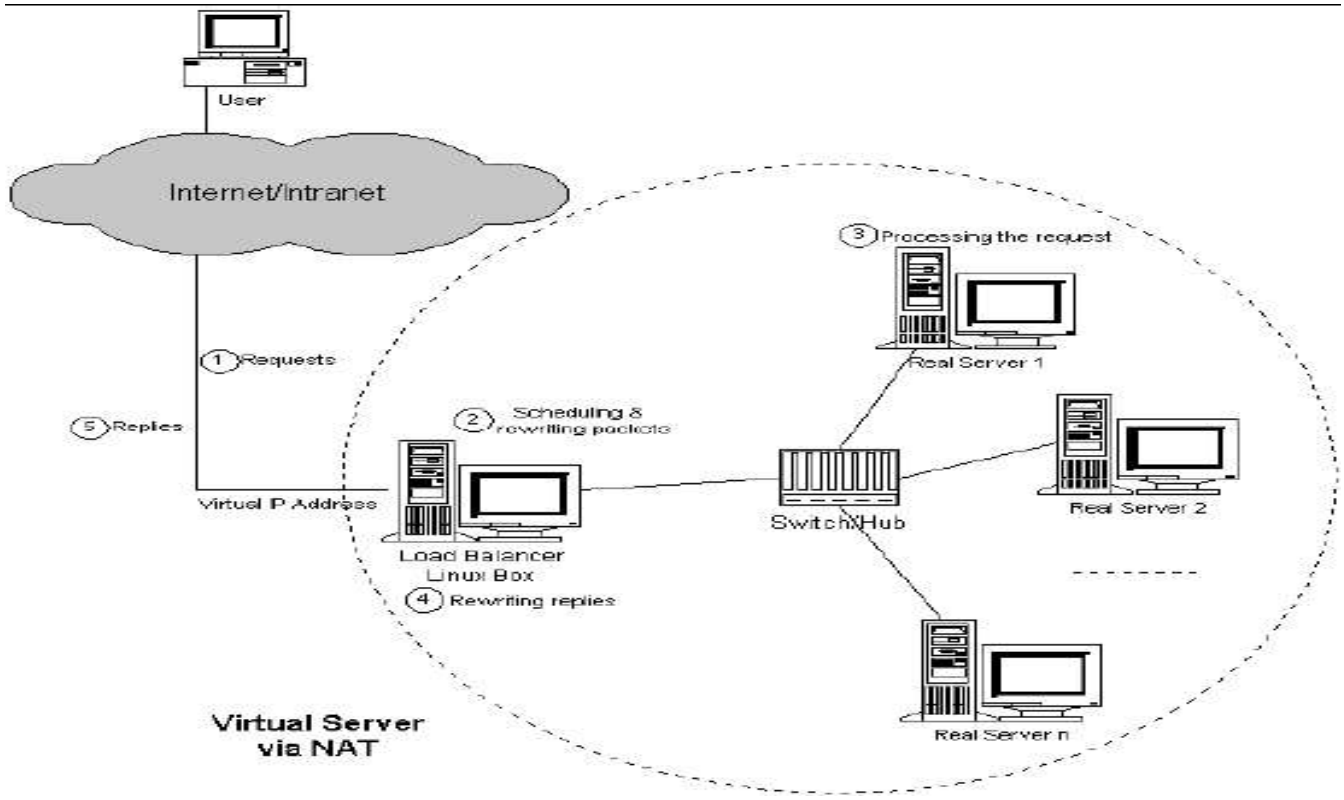


Figura 2.1 Arquitectura de Linux Virtual Server mediante NAT.

En la figura 2.1 se puede observar una configuración típica de Linux Virtual Server a través de NAT en la que los servidores reales y el balanceador de carga se hallan interconectados a través de un Switch/Hub utilizando una topología de estrella. Esta es una configuración que utiliza en forma eficiente el ancho de banda disponible en la red al separar los datos a enviarse a los servidores reales en sus respectivos canales. El reenvío de los paquetes provenientes de los clientes mediante NAT se realiza modificando los campos de dirección IP y puerto de destino de los mismos por los correspondientes a la configuración del servidor real escogido para recibir la petición. Una vez que los servidores reales procesan la petición, envían su respuesta al cliente. Debido a que los servidores reales en esta configuración utilizan como Gateway al balanceador de carga, la respuesta será enviada a través de este. El mismo entonces rescribe la dirección y puerto de origen de manera de que coincidan con los correspondientes al servicio virtual y lo reenvía al cliente. La configuración de un servidor virtual mediante NAT es muy transparente, ya que no requiere modificaciones ni

configuraciones especiales en los sistemas operativos de los servidores reales, lo cual facilita su implementación. Como desventaja de esta técnica se puede mencionar la restricción de que los servidores reales se deban encontrar en la misma red que el balanceador de carga, sin embargo esto normalmente no representa un impedimento grave dado que esta es una configuración estándar. Sin embargo, la principal desventaja de esta técnica es su escalabilidad, debido al overhead en que se incurre al tener que rescribir todos los paquetes de datos originados por los servidores reales que se hace notorio cuando el número de estos últimos es muy grande. Este efecto adverso puede disminuirse utilizando un equipo más potente como redirector. La solución a este problema consiste en que los servidores reales envíen los paquetes de datos de salida directamente hacia los clientes sin pasar por el redirector a través de las técnicas que se describen en las siguientes secciones. Visto de una forma más “física”, el montaje del clúster quedaría así, (Ver Anexo 2.1). [1]

2.2 lvs mediante IP Tunneling

Esta técnica consiste en encapsular datagramas IP en otros datagramas IP. Cuando un cliente realice una petición hacia el balanceador este comprobará el puerto e IP de destino, si estos se encuentran en la tabla definida en el LVS el balanceador encapsulara el paquete en otro datagrama IP con destino al servidor real definido en la tabla de LVS. El paquete llegará al servidor real y este lo desencapsulara y procesara la petición. El servidor responderá directamente al cliente sin necesidad de volver a pasar por el balanceador a diferencia del uso de NAT. La principal ventaja de este método es que los servidores reales pueden estar en ubicaciones físicas distintas con lo que la escalabilidad del servicio es muy superior. También

disminuye notablemente la carga en el balanceador respecto al método anterior

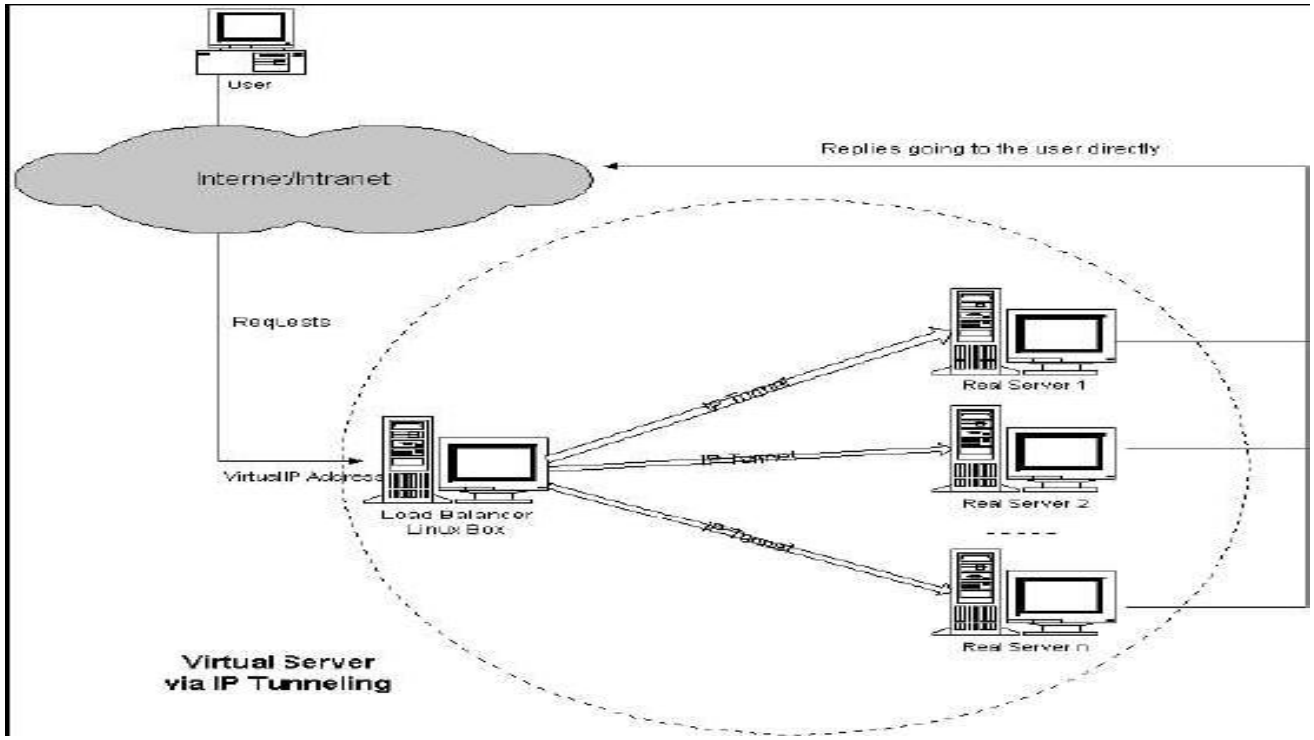


Figura 2.2 Arquitectura de Linux Virtual Server mediante IP Tunneling.

En la figura 2.2 se muestra la arquitectura de Linux Virtual Server utilizando la técnica de IP Tunneling. Para el reenvío de los paquetes, el redirector encapsula los datagramas con dirección IP del cliente como origen y destino en la dirección IP del servicio virtual dentro de datagramas con dirección IP del redirector como origen y dirección IP del servidor real como destino. Estos nuevos paquetes pueden dirigirse directamente hacia los servidores reales o ser reenviados su dispositivo de Tunneling, así que procesa el datagrama y luego envía la respuesta directamente hacia el cliente. La configuración de Linux Virtual Server a través de IP Tunneling es la más flexible respecto a la configuración física de la red, no se tienen restricciones en cuanto a la ubicación física de los servidores reales, los cuales podrían no encontrarse en la misma red local. Por otro lado, esta técnica impone la restricción de que los sistemas operativos de los servidores reales deben soportar el protocolo IP Tunneling y estar configurados apropiadamente. Utilizando esta técnica se evita el overhead que representa realizar NAT en el redirector para los paquetes de salida. Esto multiplica la escalabilidad del

sistema para prácticamente todas las aplicaciones cliente/servidor ya que normalmente el volumen de los datos que envía el cliente al servidor es significativamente menor con respecto a los que el servidor envía al cliente. De esta forma el límite en cuanto a cantidad de servidores reales que pueden agregarse al sistema obteniendo un aumento lineal de rendimiento es mucho más alto que el que se tiene utilizando NAT. [1]De una forma muy genérica, el encapsulado funciona así, (Ver Anexo 2.2).

2.3 lvs mediante Direct Routing

Direct Routing es una técnica en la que la dirección IP virtual del balanceador es compartida tanto por los servidores reales como por el balanceador. El balanceador tendrá 2 direcciones IP sobre el mismo interfaz, una la propia IP del servidor y otra con la IP virtual. Los servidores reales tendrán la IP virtual configurada sobre el loopback ya que esta interfaz no responde a peticiones ARP y así evitaremos que n servidores más el balanceador atiendan a las peticiones ARP. Tanto el balanceador como los servidores reales deben compartir el mismo segmento de red y deben estar interconectados mediante un hub o switch. El balanceador recibirá la petición del cliente con destino a la IP virtual, entonces este comprobará la IP y puerto de destino sobre su tabla de servidores reales. El balanceador en base a esto enviará la petición directamente al servidor correspondiente y este último responderá directamente al cliente que realizo la petición original. El balanceador simplemente cambiará la MAC dentro del frame y lo reinyectara de nuevo en la LAN. La gran ventaja de esta técnica es la baja carga que recibirá el balanceador, incluso inferior al del LVS tunelizado al tener que realizar una tarea sencilla como es la modificación de la MAC que utiliza menos recursos de procesamiento respecto al reencapsulado de los datagramas IP.

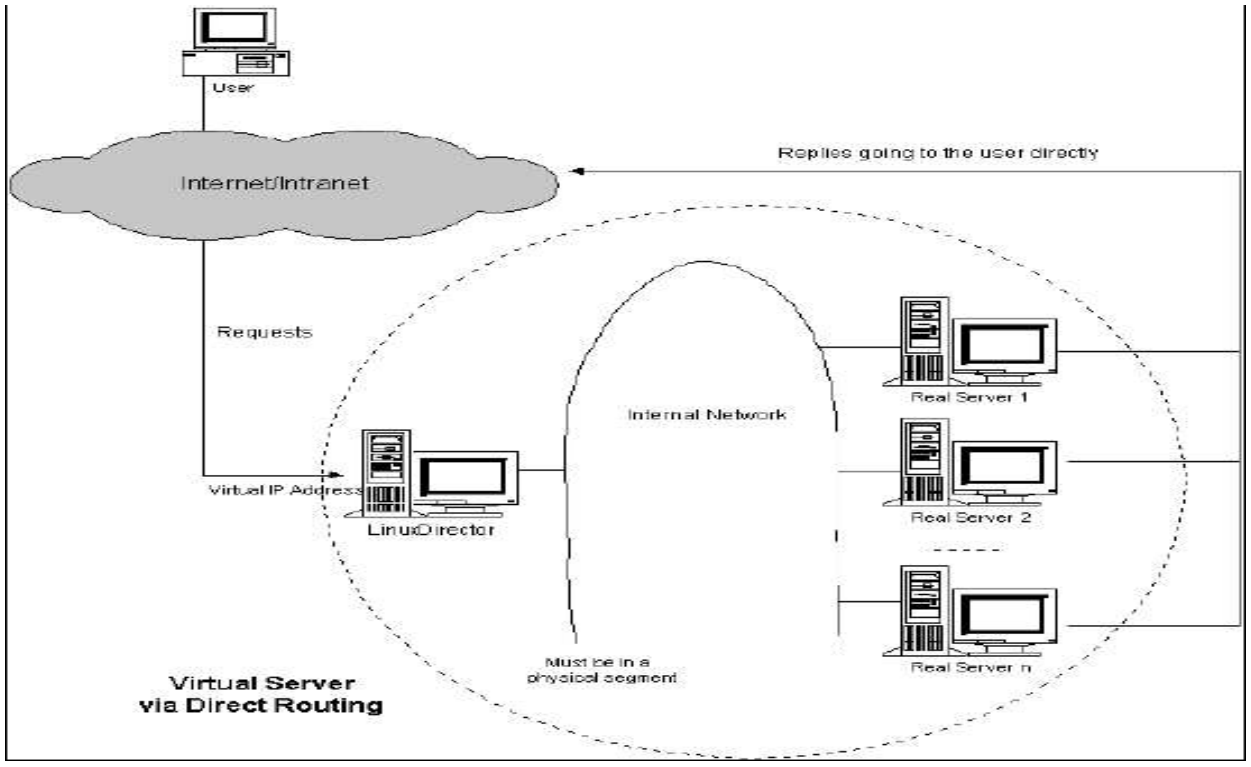


Figura 2.3 Arquitectura de Linux Virtual Server mediante Direct Routing.

Para realizar el reenvío de los paquetes, el balanceador de carga sólo cambia las direcciones MAC para direccional los paquetes al servidor real elegido mientras que la dirección IP de destino sigue siendo la del servicio virtual. Los servidores reales pueden aceptar los paquetes IP entrantes ya que tienen un dispositivo con esa dirección IP asignada. Luego procesan el requerimiento y envían la respuesta directamente hacia el cliente. La configuración de Linux Virtual Server a través de Direct Routing impone la restricción de que los dispositivos de red que tengan asignada la dirección del servicio virtual deban no responder a las peticiones de ARP, lo cual puede no ser un problema trivial en algunos sistemas operativos (curiosamente esto se da en los kernels 2.2.x y 2.4.x de Linux Por otro lado esta técnica, al igual que la de IP Tunneling, multiplica la escalabilidad del sistema al descargar del balanceador de carga la tarea de redireccional los paquetes de datos de los servidores a los clientes, pero como ventaja adicional con respecto a IP Tunneling cuenta con el hecho de que se elimina el overhead derivado del encapsulamiento/desencapsulamiento de los datagramas IP. Esta tecnología se utilizó para proporcionar servicios http en las olimpiadas de Atlanta y de Sydney,

por lo que destaca como una solución capaz de servir un gran volumen de peticiones. También se utilizó para retransmitir vía web la partida de ajedrez entre Kasparov y la máquina Deep Blue.En la siguiente tabla se resumen las características principales de los tres métodos de direccionamiento que puede utilizar el balanceador de carga de Linux Virtual Server:

	NAT	IP tunneling	Direct Routing
Servidor	cualquiera	necesita encapsulamiento	dispositivo no-ARP
Red de servidores	red privada	LAN/WAN	LAN
Escalabilidad	baja (10~20)	alta	alta
Salida hacia Internet	balanceador	router	router

Figura 2.4 LVS: Métodos de direccionamiento

2.4 Algoritmos de scheduling

Los balanceadores de carga pueden utilizar los métodos explicados anteriormente para decidir a qué servidor asignar las diferentes conexiones. Algunos de ellos son más efectivos que otros, en función del servicio que se ha de balancear. Es por esto, que un balanceador de carga puede utilizar diferentes algoritmos de scheduling para diferentes tipos de servicios.

Round-Robin (rr): El algoritmo de Round-Robin envía cada petición de entrada al siguiente servidor de la lista. Por ejemplo, en el supuesto de tener tres servidores, A, B y C, la primera petición la recibiría el servidor A, la segunda el B y la tercera el C. El ciclo se vuelve a repetir empezando nuevamente por el servidor A. Virtual server proporciona algunas ventajas sobre el tradicional Round-Robin DNS.

Distribución Ponderada por pesos (Weighted Distribution**)**: Este método permite al administrador asignar un peso a cada servidor, en función de su capacidad y el servicio que ha de prestar. Se utilizan en combinación con Round-Robin y Least-Connections.

Weighted Round-Robin (wrr): Este algoritmo está diseñado para mejorar el manejo de servidores con diferentes capacidades de proceso. A cada servidor se le puede asignar un peso, que es un valor entero que indica la capacidad de proceso y que determina la proporción del trabajo que le será asignado. A mayor peso, se le envían más peticiones. Weighted Round-Robin es mejor que Round-Robin cuando la capacidad de proceso de los servidores reales es diferente. Actualmente, el Round-Robin es un caso particular del Weighted Round-Robin, en el cual todos sus pesos son iguales.

Least-Connection (lc): Dirige las conexiones de red al servidor con el menor número de conexiones. Este algoritmo es de naturaleza dinámica ya que necesita contar las conexiones activas para cada servidor dinámicamente, y es adecuado en el caso de que la carga varía notablemente.

Weighted Least-Connection (wlc): El Weighted Least-Connection scheduling es un perfeccionamiento del Least-Connection en el cual se le puede asignar un peso variable a cada servidor real.

Locality-Based Least-Connection (lblc): Asigna trabajos destinados para la misma IP para el mismo servidor, si el servidor no está sobrecargado y disponible. En caso contrario, lo asigna al servidor con menos trabajo y lo mantiene para designaciones futuras.

Locality-Based Least-Connection with Replication (lblcr): Asigna trabajos destinados para la misma IP al servidor Least-Connection configurado para esa IP. Si todos los servidores están sobrecargados, repone al servidor con menos trabajo y lo añade a la lista de disponibles.

Destination Hashing (dh): Asigna trabajos a los servidores a base de buscar una asignación estática, por la IP destino, en la tabla hash.

Source Hashing (sh): Asigna trabajos a los servidores a base de buscar una asignación estática, por la IP origen, en la tabla hash.

Por tiempo de respuesta (Response Time): Complementa a otros algoritmos posibilitando la elección del servidor más rápido. [2]

2.4.1 Los cuatro más utilizados

Entre estos 10 algoritmos pues los más utilizados, los más interesantes, los más efectivos son 4 que seguidamente se detallarán con más características que las expresadas anteriormente para reconocer mejor cual puede ser utilizado para la propuesta de solución:

- **Round-Robin.** Este algoritmo es el más simple y lo que hace es distribuir las peticiones entre los servidores de tal manera que si hay 5 servidores y 100 peticiones cada servidor atenderá a 20 peticiones.

  El orden de distribución de la carga será secuencial, primera petición hacía el primer servidor, segunda al segundo, .., quinta al quinto, sexta al primero, ...

  Esta distribución es muy sencilla pero presupone que todas las peticiones van a ser equivalentes, en términos de carga, para el servidor, algo que en la realidad dista mucho de ser cierto. O que la capacidad de procesamiento de los servidores es la misma.

  Podría darse el caso de que haya servidores atendiendo varias peticiones y otros esperando o que los servidores más lentos estuvieran muy sobrecargados mientras que los más potentes estuvieran más desahogados. [6]

- **Weighted Round-Robin.** Este algoritmo permite un aprovechamiento mejor del clúster cuando hay máquinas con diferentes capacidades de procesamiento, de esta forma a las máquinas con mayor capacidad de procesamiento se les dará una mayor prioridad (weight) para responder a las peticiones de los clientes y el balanceador distribuirá la carga entre los servidores teniendo en cuenta su prioridad. [6]

En realidad el Round-Robin Scheduling es un Weighted Round-Robin Scheduling y todas las prioridades son iguales para los servidores

- **Least-Connection.** Con este algoritmo las peticiones se enviaran al servidor que menos conexiones este sirviendo en ese momento.

  Si la capacidad de procesamiento de los servidores es similar este algoritmo distribuirá la carga de forma óptima entre todas las máquinas del clúster. Sin embargo si las capacidades de procesamiento varían mucho la carga no será repartida de forma ecuánime ya que la carga se repartirá según el número de conexiones abiertas en ese momento y no sobre la carga real de cada máquina. [6]

- **Weighted Least-Connection.** Este algoritmo es al Least-Connection scheduling lo que el weighted Round-Robin Scheduling es al Round-Robin Scheduling. A cada servidor se le asigna una prioridad según su capacidad de procesamiento y aquellos que mas prioridad tengan asignada atenderán más peticiones, es decir tendrán más conexiones abiertas.[6]

2.5 Como distribuir la carga

Existen varias formas para montar un clúster y distribuir la carga entre los equipos. El método más sencillo es mediante un DNS round-robin: Cuando en un servidor de DNS definimos varias IP para un mismo dominio, el servidor nos devolverá cada vez una IP distinta, en principio sin ningún criterio en especial (no está especificado), aunque generalmente se utiliza una cola round-robin para ir sirviendo las peticiones. De esta forma conseguiríamos distribuir la carga entre los servidores reales de una forma pseudo-aleatoria, según vayan llegando peticiones. El problema con este método es que no se tiene en cuenta la carga real de cada servidor, y es posible que todas las peticiones “pesadas” vayan a parar siempre a la misma máquina que acabaría por saturarse, mientras que las demás estarían sirviendo peticiones triviales. Otro problema es que como los clientes suelen mantener una caché de los dominios ya resueltos a través del DNS, una vez que un cliente haya contactado con uno de los servidores reales, siempre (hasta que expire su caché) se dirigirá al mismo servidor. Este es

otro punto por el que se puede llegar a sobrecargar un servidor mientras que el resto están libres. Además, si en algún momento fallara un servidor y su IP está todavía en la caché de algún cliente, éste seguiría enviándole las peticiones que, por supuesto, fallarían. Una solución mejor es utilizar un balanceador de carga para distribuir las conexiones entre los servidores. Con esta solución se puede aumentar la sensación de "unidad" del clúster, ya que de cara al usuario únicamente habrá una dirección IP a la que se dirijan todas las peticiones, en lugar de varias. La granularidad de la distribución se puede hacer por conexión (cada petición de un cliente se redirige al servidor que esté en mejor posición para servirlo) o por sesiones (se almacena en una tabla a qué servidor se envía a cada cliente y se le manda siempre al mismo). Cuando algún servidor falla, es más fácil enmascarar el error, ya que únicamente habrá que proveer al balanceador de carga de los mecanismos necesarios para detectar el fallo de un servidor y eliminarlo de su lista, de forma que no le redirija ninguna petición. Por este mismo motivo, la administración del clúster se simplifica ya que se pueden sacar servidores del clúster en cualquier momento para realizar tareas de mantenimiento, sin que ello provoque errores en los clientes. El balanceo de carga se puede hacer a dos niveles: a nivel de conexión IP, o a nivel de protocolo. En este segundo caso, el balanceador sería una especie de proxy que programaríamos para recibir conexiones en un determinado puerto, tal vez inspeccionar los paquetes para ver si se trata del protocolo correcto o extraer algún tipo de dato del protocolo e incluso poder filtrar peticiones incorrectas, y redireccionar la conexión hacia uno de los servidores. El problema de esta aproximación es que al tener que analizar el protocolo en todas las conexiones entrantes, el programa balanceador es bastante complejo y podría llegar a convertirse en un cuello de botella en la entrada al clúster (se estima que, dependiendo de la potencia de los equipos, el número de servidores reales que se pueden servir sin problemas de congestión estaría entorno a los 4-6). Entre este tipo de balanceadores contamos con Reverse-Proxy y pWEB. La otra opción, el balanceado a nivel de conexión IP, es mucho más eficiente ya que el proceso a realizar es también mucho más sencillo: cuando llega una petición al balanceador no se analiza a ningún nivel mayor que el de TCP/IP, lo justo para aceptar la conexión y redirigirla a uno de los servidores. Con esta aproximación los servidores reales que se pueden tener detrás del balanceador pueden

oscilar entre 25 y hasta 100, como siempre, según la potencia de los equipos. El balanceador de Linux Virtual Server funciona a este nivel. [2]

2.6 Algunas Herramientas de Balanceo de Carga

Por otra parte, han surgido varias herramientas más o menos gráficas y más o menos cómodas para realizar de forma interactiva todo el proceso de configuración pero están en dependencia del sistema operativo que se tenga instalado en la PC ejemplos de ellas son:

Lvs-gui: Es una aplicación gráfica para X-Windows desarrollada por VA Linux para configurar fácilmente un clúster LVS. Lvs-gui nos permite configurar de forma remota (realiza la conexión mediante ssh) tanto el equipo que haga de balanceador de carga como el resto de servidores. El programa se encarga de, tras configurar el clúster mediante su interfaz gráfica, conectarse a los equipos remotos y modificar sus ficheros de configuración para que implementen con nuestra configuración. [3]

LVSM: Linux Virtual Server Manager es otro programa que nos permitirá de una forma cómoda y fácil instalar y administrar un clúster LVS a través de una interfaz en HTML. LVSM está programado en Perl, y necesita de Apache con el módulo mod_perl para funcionar. LVSM nos ofrece, Balanceo de carga, Alta disponibilidad, Monitorización de servicios, Administración remota de servidores web, Configuración sencilla. [3]

Estadísticas del clúster. Ultra Monkey: sería la solución más rápida y fácil si necesitamos montar un clúster de servidores web. Se trata de un paquete con TODO el software necesario para montar un clúster LVS: un kernel con los parches adecuados, mon, heartbeat, fake ... Además de todo el software, también ofrece abundante documentación sobre cómo configurar el clúster y varios ficheros de configuración ya preparados. [3]

Super Sparrow: Super Sparrow lleva el concepto del clustering y el balanceo de carga un paso más allá, permitiéndonos hacer un “clúster de clústeres” distribuidos geográficamente, de forma que cada cliente se encamine hacia el clúster que le resulte más cercano (en términos del enrutamiento a través de Internet) y que, por tanto, debería irle más rápido. Además, es un

gran ejemplo de ingenio, ya que tomando como base un protocolo existente, lo reaprovecha con éxito para un fin distinto a aquel para el que fue diseñado. Visto de otra forma, Super Sparrow viene a automatizar en la parte del servidor el proceso de elegir el “mirror” de una determinada web más próxima a nuestra localización, en lugar de tener que seleccionarlo manualmente de una lista. [3]

2.4.1 Piranha:

Es también una solución completa, al igual que Ultra Monkey, de la mano de Red Hat. Se basa en LVS, algunos programas GPL, y es una implementación completamente software. Piranha es tanto el nombre del producto, el clúster en sí, como del interfaz gráfico para administrarlo. Un clúster Piranha se compone de los siguientes elementos:

-El parche IPVS para el kernel.
-El demonio lvs para manejar las tablas IPVS a través de ipvsadm.
-El demonio nanny para monitorizar servicios y servidores.
-El demonio pulse para controlar el estado del resto de demonios del clúster y la entrada en -funcionamiento del distribuidor de carga de respaldo en caso de fallo del primario.
-La interfaz gráfica piranha para administrar el clúster.

Todos estos programas utilizan el mismo fichero de configuración, /etc/lvs.cf. La función de la interfaz gráfica piranha es la de iniciar o detener el demonio pulse de forma interactiva, y editar los parámetros de configuración. El código IPVS de Linux Virtual Server es el encargado de la distribución de carga entre los servidores, estando disponibles todos los métodos comentados anteriormente. Como valor añadido a todo lo comentado para una instalación LVS normal, piranha es capaz de adaptar los pesos de los algoritmos de planificación automáticamente, según las estadísticas de funcionamiento de cada servidor.Todo servicio prestado por cada uno de los servidores reales de nuestro clúster se monitoriza mediante el demonio nanny, en ejecución en el distribuidor de carga activo.

Esta monitorización se lleva a cabo en dos pasos: en primer lugar se comprueba el estado de las tarjetas de red y de la propia red en sí, y se trata de conectar con el servidor para

asegurarnos de que no se ha colgado la máquina; en segundo lugar, se establece una conexión al puerto del protocolo que estemos monitorizando, se envía una petición sencilla y se espera la respuesta correspondiente. Este proceso se repite cada dos segundos. Si un servidor no responde (o la respuesta no es la esperada) durante un cierto período de tiempo, nanny se encarga de eliminarlo de las tablas de IPVS para así "sacar" el servidor caído del clúster. El equipo con problemas sigue siendo monitorizado, para reinsertarlo en el clúster de forma automática cuando vuelva a funcionar.Para que el balanceador de carga no sea un SPF, se puede montar un segundo balanceador que le haga de respaldo en caso de problemas. Cuando se configura piranha de esta forma, el balanceador de respaldo mantiene una copia de la configuración del clúster y monitoriza el estado del balanceador primario. Si se detecta algún problema durante un cierto tiempo, el balanceador de respaldo lleva a cabo las acciones necesarias para suplantar la IP del balanceador principal y tomar su lugar. Si, tras un tiempo, el antiguo balanceador vuelve a funcionar, detecta que el otro ha tomado su lugar y a partir de entonces actuará como balanceador de respaldo, monitorizando el estado del nuevo balanceador principal. [3]

Conclusiones

Después de leer toda la documentación del capítulo se puede definir la técnica y el algoritmo a utilizar que es la respuesta a la pregunta científica del comienzo, Además también se destaca como se puede distribuir la carga y se describen las herramientas a utilizar donde se elige Piranha debido a que es una magnifica solución y es la que esta implementada para el sistema Red Hat que es el que tiene instalado en nodo central de la uci en los servidores que se les va aplicar el balance de carga. La Técnica finalmente escogida es Direct-Routing (Ruteo Directo) porque utilizando NAT teníamos un cuello de botella en el balanceador ya que tiene que reescribir y distribuir los paquetes del cliente al servidor y viceversa por lo que es de baja escalabilidad y además de que solo se usa para redes pequeñas, privadas. Utilizando IP Tunneling el balanceador únicamente tendrá que hacerse cargo de las peticiones de los clientes y enviarlas a los servidores, siendo estos mismos los que responderán a los clientes.

De esta forma el balanceador de carga puede manejar más nodos, es decir el servicio es más escalable pero la ventaja adicional con respecto a IP Tunneling es que cuenta con el hecho de que se elimina el overhead derivado del encapsulamiento/desencapsulamiento de los datagramas IP. Por último el algoritmo de sheduling que se escogió fue el Weighted-Least-Connection **(WLC)** ya que es una combinación del Least Connection con el Weighted Distribution y pues se pueden utilizar las ventajas de los 2 en uno solo y su principal virtud es asignar un peso variable a cada servidor real.

CAPITULO 3: ¿COMO APLICAR LINUX VIRTUAL SERVER (LVS)?

Introducción

Para poder utilizar todos los servicios que brinda LVS primeramente se deben instalar y configurar una serie de herramientas que posibilitan el desarrollo exitoso de este balanceador de carga, en el caso específico del sistema operativo de Red Hat, pues tratamos con la herramienta de piraña como se describe anteriormente pero además debemos agregar el ipvsadm, otros sistemas operativos como el caso de Debian o Ubuntu pues utilizan el Ultramonkey o el Keplive junto con el ldirector o el Webmin que pueden realizar todo ese trabajo. Además se debe tener en cuenta la seguridad y capacidad de trabajo de los servidores y cuanto hardware se necesita para balancear la carga. Por tanto este capítulo traerá consigo una guía de cómo realizar paso a paso todo el trabajo que facilitara a cualquier informático usar en caso de ser necesario el LVS específicamente con la vía de solución propuesta.

3 Pasos para la instalación

Antes que nada se debe tener instalado el sistema operativo y en el caso de la solución planteada pues sería Red Hat, seguidamente descargar e instalar el kernel 2.4.5 o 2.6.10 o posterior, y descargar la utilidad ipvsadmn para administrar el interior del núcleo de Linux. El software de LVS se divide en dos partes, un parche para el kernel del equipo que vaya a hacer de balanceador (IPVS-patch); y unas herramientas de administración (ipvsadm), que también se utilizarán desde el balanceador. En los servidores, únicamente tendremos que instalar y configurar el software servidor que vayamos a necesitar, y configurar la red (IPs, Gateway, encapsulado IP...) según el tipo de direccionamiento que vayamos a usar en el clúster (VS-NAT, VS-Tun o VS-DR).Tras parchear el kernel, en make menuconfig nos aparecerá una nueva serie de opciones para habilitar LVS. Una vez tenemos parcheado, compilado y funcionando el kernel con soporte IPVS en el distribuidor, tendremos que configurar el clúster con ayuda del programa en línea de comandos ipvsadm es la herramienta para administrar el Servidor Virtual Linux (LVS) que permite configurar, mantener y visualizar

el estado de la tabla del servidor virtual del kernel de Linux. Así, mediante la interfaz de usuario ipvsadm, es posible configurar los servicios y servidores que direccionará IPVS, el peso asignado a cada servidor, útil cuando unos servidores son más rápidos que otros, y el algoritmo de scheduling utilizado para elegir el servidor real al que deberán reenviarse los paquetes tratados por el balanceador. Ipvsadm puede ser invocado desde la línea de comandos o utilizado para crear scripts de inicialización que definan el comportamiento de IPVS.

Podemos utilizar ipvsadm para realizar las siguientes funciones:
- Añadir servicios
- Desactivar servicios
- Borrar servicios
- Lanzar o detener el demonio de sincronización de estado
- Salvar la configuración actual del LVS (ipvsadm-save)
- Restaurar una configuración para el LVS (ipvsadm-restore)

3.1 Como utilizar ipvsadm

Todas estas opciones se ejecutan a través de comandos pero antes se debe activar el ipvsadm que sería un script de esta manera. /etc/inid.t/ipvsadm start. El formato de ejecución de ipvsadm es el siguiente:

Ipvsadm COMANDO PARAMETROS
Que en función del comando (MAYÚSCULAS/minúscula) da lugar a dos formatos básicos de ejecución:
- *ipvsadm COMANDO [protocolo] dirección-del-servicio [método-de-scheduling] [opciones de persistencia]*
- *ipvsadm comando [protocolo] dirección-del-servicio dirección-del-servidor [método de reenvío] [peso].*

El primero de ellos define los servicios del LVS y el algoritmo utilizado para asignar, a un servidor real determinado, las peticiones a esos servicios. Opcionalmente, si el servicio

requiere de persistencia, es posible definir el timeout y la máscara de red. El segundo formato de instrucción, especifica los servidores reales que han de ser asociados a cada uno de los servicios virtuales prestados por el LVS. Cuando se añade un servidor real, el método de reenvío debe de ser especificado o de lo contrario se tomarán los valores por defecto. Si se ejecuta ipvsadm sin especificar ningún comando, el programa lista los servicios configurados y los servidores reales que están configurados para prestarlos. Lo que equivale a llamar a ipvsadm con el comando –L ó –l. [2]

3.1.2 Comandos en Mayúscula

-A, --add-service

Añade un servicio virtual que queda definido únicamente por la tripleta: Dirección IP, número de puerto y protocolo.

root@director1:~# ipvsadm –A –t 10.31.16.30: 80 –s wlc

Este comando configura el balanceador de carga con el servicio http, y de esta manera IPVS se hará cargo de todos los paquetes destinados a la IP 10.31.16.30 dirigidos al puerto 80 (http). Posteriormente con ipvsadm –a, se han de añadir los servidores reales a los que se reenviarán las peticiones al servicio que se acaba de configurar.

-D, --delete-service

Este comando elimina un servicio virtual así como cualquier servidor real asociado y debe de ir acompañado del uno de los siguientes parámetros -t | -u | -f (-t para TCP, -u para UDP y -f para firewall-mark).

Para borrar el servicio http y sus servidores asociados, se usará la instrucción con el comando anteriormente descrito:

root@director1:~# ipvsadm -D -t 10.31.16.30: http

-C, --clear

Borra todas las entradas de la tabla de servicios del servidor virtual y sus respectivos realservers, pero **NO** elimina las entradas de la tabla de conexiones establecidas. [2]

3.1.3 Comandos en Minúscula

-a, --add-server

Permite añadir un servidor real. De esta manera se le puede indicar al balanceador a que servidor real debe de enviar los paquetes pertenecientes a un servicio configurado previamente con el parámetro –A.

root@director1:~# ipvsadm -a -t 10.31.16.30:80 -r 10.31.20.122 –m

-d, --delete-server

Permite borrar un servidor real, con lo que dejan de reenviársele paquetes

-l, -L, list

Lista la tabla de servidores virtuales mostrando todos los servicios configurados en ellos. Si además se especifica una service-address, se mostrarán únicamente las entradas para este servicio.

--start-daemon state

Lanza el demonio de sincronización de estado que está implementado dentro del kernel de Linux. Este comando es utilizado cuando se implementa LVS en un entorno de alta disponibilidad en una arquitectura de clustering activo/pasivo. El demonio permite tener varias máquinas parcheadas con IPVS actuando como balanceadores de carga. [2]

--stop-daemon

Detiene el demonio de sincronización de conexiones

-h, --help

Ayuda, muestra una descripción de la sintaxis de los comandos

3.2 Parámetros

Los comandos mostrados hasta el momento, requieren o aceptan algunos de los parámetros que se listan a continuación:

-t, --tcp-service service-address

Se utiliza para especificar un servicio TCP Service-address se usa en la modalidad host[:port]. Pudiéndose especificar el servidor real con su dirección IP o bien con su nombre host (tal y como aparece en /etc/hosts). La especificación del puerto es opcional, si se omite se utiliza cero por defecto, lo cual sólo es válido si el servicio es persistente (opción -p). La opción –p y un cero en el nombre de puerto, configura al balanceador para realizar Persistent Client Connection (PCC) en vez de Persistent Port Connection (PPC).

-u, --udp-service service-address

Usado para especificar un servicio UDP. La dirección especificada en service-address sigue los mismos convenios explicados para el parámetro –t del apartado anterior.

-f, --fwmark-service integer

Permite utilizar una marca de firewall (entero mayor que 0) en lugar de una dirección, puerto y protocolo (TCP ó UDP), para designar un servicio virtual. Para marcar los paquetes, estos deberían procesarse previamente con iptables (opción -m). Las marcas de firewall pueden ser utilizadas para asociar diferentes servicios virtuales (port grouping). De manera que una sola definición de servicio virtual en IPVS, cubra diferentes servicios virtuales correspondientes a múltiples tripletas: dirección, puerto y protocolo; siempre que estos servicios estén ofrecidos, todos ellos, en los mismos servidores reales. De lo contrario, se podrían enviar peticiones a servicios no ofrecidos en el servidor destino.

De esta forma, las firewall-marks permiten agrupar convenientemente diferentes direcciones IP, puertos y protocolos bajo un solo servicio virtual. Lo cual simplifica la configuración cuando son requeridos un elevado número de servicios y se necesita agrupar la persistencia, que de otro modo requeriría de múltiples servicios virtuales

-s, --schedules scheduling-method

Parámetro usado para especificar el algoritmo utilizado por el scheduler para asignar a un servidor real los datagramas UDP y las conexiones TCP. Los algoritmos se implementan como módulos del kernel. Actualmente se pueden utilizar los siguientes métodos de

scheduling: rr Round Robin wrr Weighted Round Robin lc Least-Connection wlc Weighted Least-Connection lblc Locality-Based Least-Connection lblcr Locality-Based Least-Connection with Replication dh Destination-Hashing sh Source-Hashing.

-p --persistence [timeout]

Especifica qué servicios virtuales son persistentes. Si esta opción es especificada, las múltiples peticiones de un cliente son redireccionadas hacia el mismo servidor real seleccionado para la primera petición. Opcionalmente, el timeout de las sesiones persistentes permite ser cuantificado en segundos por el usuario, de lo contrario, se utiliza el valor por defecto que es de 300 segundos. Esta opción es utilizada en aquellos protocolos, como SSL o FTP, en los que es importante que los clientes conecten constantemente con el mismo servidor real.

Nota: Si un servicio virtual ha de manejar conexiones FTP, entonces requiere ser configurado para aplicar reglas de persistencia para el servicio, en el caso de que se utilice Direct Routing como mecanismo de reenvío.

-r --real-server server-address

Las peticiones a un servicio virtual han de ser asociadas a un servidor real. En este sentido, el algoritmo de scheduling seleccionará un servidor real de entre los que especifiquen con ese parámetro. El parámetro server-address es la dirección del servidor real, y también permite especificar un número de puerto. La dirección del servidor real (server-address) puede proporcionarse mediante la dirección IP o el nombre de host. El puerto puede indicarse también mediante el número de puerto o bien mediante el nombre de servicio para ese puerto (http, ftp, etc).

-g –gatewaying (Direct Routing), -i --ipip(Tunneling), -m --masquerading (NAT).

-w, --weight weight

El peso (weight) es un número entero que especifica la capacidad relativa de un servidor real frente al resto de servidores reales del clúster. Los valores válidos para este van de 0 a 65535, pero el valor por defecto es 1. Se puede especificar un servidor inactivo asignándole un peso igual a cero. Un servidor inactivo no recibirá nuevas peticiones, sea cual sea el método de scheduling del LVS, pero seguirá ofreciendo sus servicios. Configurar un servidor inactivo

puede utilizarse para evitar la sobrecarga de un determinado servidor real, o bien para dejarlo fuera de servicio y poder realizar labores de mantenimiento. [2]

--mcast-interface interface

Especifica la interfaz por la cual el demonio de sincronización de conexiones enviará o recibirá los mensajes multicast. Este parámetro se utiliza junto a --start-daemon, que lanza el demonio de sincronización de conexiones.

root@director1:~# ipvsadm –-start-daemon=master –-mcast-interface=eth1

3.3 Trabajo con la herramienta Piranha

Con la herramienta piranha-gui pues nos ahorramos mucho trabajo ya que es un entorno web muy amigable y favorable debido a que es cómodo de trabajar además de que no se tiene que estar personalmente en la estación de trabajo, ya que con una cuenta de administración pues es capaz desde otra pc pues poder revisar y configurar el LVS. Con solo seguir los pasos mencionados y mirar el entorno web pues resulta sencillo un trabajo que puede ser un poco tedioso. Esta sección provee una visión general de la herramienta de configuración LVS disponible, Piranha Configuration Tool . La herramienta de Piranha Configuration es una interfaz gráfica del usuario de navegador de internet que le provee un acercamiento estructurado. Para acceder a la herramienta de Piranha Configuration se necesita el servicio de piranha-gui funcionando con el LVS activo. Se puede acceder a la herramienta de Piraña Configuration localmente o remotamente con un navegador de I\internet. Además se puede acceder a ello localmente con este URL: El http://localhost:3636. Tambien puede acceder a eso remotamente con ya sea el hostname o la dirección verdadera IP seguido: 3636. Si se accede a la herramienta remotamente, se necesita una conexión del ssh para activar LVS router como el usuario root o el administrador. Echar a andar la herramienta de Piraña Configuration causa que la página bienvenida Tool Configuration Piranha sea exhibida. (Ver anexo 3.1). [5]

Entrar al sistema para la página bienvenida le provee el acceso a los cuatro paneles o pantallas principales: El (control/monitoring, (Global/settings), (Redundancy), (Virtual Server). (Anexo 3.2).

Además, el panel VIRTUAL de SERVIDORES contiene a cuatro subdivisiones. El panel de control /monitoreo es el primer panel exhibido después de que usted entra al sistema en la pantalla bienvenida. Las siguientes secciones proveen una descripción breve de las páginas de configuración de la Herramienta de la Piraña Configuration. El Panel de *control /monitoring* exhibe estado corrido de tiempo. Exhibe el estado del demonio pulse, la tabla de ruteo del LVS, y los procesos de LVS-spawned nanny. *Auto update* Le permite al despliegue de estado estar actualizado automáticamente en un intervalo de configuración incrustado en la frecuencia de actualización en cuadro de texto de segundos (el valor predeterminado es 10 segundos). No es recomendado que establezca la actualización automática para un intervalo menos de 10 segundos. El proceder puede dificultar reconfigurar el intervalo automático de actualización porque la página actualizará demasiado frecuentemente. Si se encuentra este asunto, simplemente dé un clic sobre otro panel y luego eche para atrás en el control /monitoreo. **Update information now:** Provee actualización manual de la información de estados **CHANGE PASSWORD:** Dando un clic sobre estas tomas del botón usted para una pantalla de ayuda con información en relación a la forma de cambiar la contraseña administrativa. El panel *GLOBALSETTINGS* es donde el administrador LVS define los detalles de sistema de redes para el router LVS. *Primary server public:* la dirección IP públicamente verdadera hurgable para el nodo primario LVS. *Primary server private IP:* la dirección verdadera IP para una interfaz alternativa de la red en el nodo primario LVS. Esta dirección es usada solamente como un canal alternativo del heartbeat para router de apoyo. Use network type Los siguientes tres campos son específicamente para la interfaz virtual de la red, conectada la red privada con los servidores verdaderos, con las topologías Nat, Direct Routing y Ip Tunneling, para la solución planteada recuerde escoger la segunda opción. (Anexo 3.3) el panel *REDUNDANCY* le permite configurar la copia de seguridad del nodo LVS y configurar diversas opciones de seguimiento heartbeat. *Redundant server public IP*: El público la dirección IP real para la copia de seguridad LVS router. El resto del grupo es para configurar el canal de heartbeat, que es utilizado por el nodo de copia de seguridad para vigilar el nodo principal por si fracasa. *Heartbeat Interval (seconds):* Establece el número de segundos entre el intervalo que el nodo de copia de seguridad comprobará el estado funcional del nodo principal LVS. *Assume death after (seconds)*: Si el nodo principal LVS no responde después

de este número de segundos, entonces la copia de seguridad LVS router nodo se pondrá en marcha en caso de caídas. *Heartbeat runs on port*: Define el puerto en el cual el heartbeat se comunica con el nodo principal LVS. El valor por defecto es establecido a 539 si dejas este campo en blanco. (Ver Anexo 3.4). [5]

VIRTUAL SERVER: pantallas de información que definen cada servidor virtual. Cada entrada en la tabla se muestra la situación del servidor virtual, el nombre del servidor, la IP virtual asignado al servidor, la máscara de red virtual de la propiedad intelectual, el número de puerto para que el servicio se comunique con el protocolo utilizado, y la interfaz de dispositivo virtual. Cada servidor está representado en el panel de Virtual servers. Para agregar un servicio, haga clic en el botón **Añadir.** Para eliminar un servicio, selecciónelo haciendo clic en el botón situado junto al servidor virtual y haga clic en el **botón** "Eliminar". Para activar o desactivar un servidor virtual en el cuadro que hacer clic en el botón y haga clic en el **(DE) ACTIVAR** botón. Después de añadir un servidor virtual, puede configurar haciendo clic en el botón a su izquierda y haciendo clic en el **botón Editar** para mostrar el virtual server subsección. (Ver Anexo 3.5). *Virtual server subsección*: este panel permite configurar un servidor virtual individual. Enlaces a las subsecciones relacionadas específicamente con este servidor virtual se encuentran en la parte superior de la página. Pero antes de la configuración de cualquiera de los incisos relacionados con este servidor virtual, completa esta página y haga clic en el **botón** Aceptar. *Name* Un nombre descriptivo para identificar el servidor virtual. Este nombre no es el nombre de host para la máquina, por lo que sea descriptivo y fácilmente identificable. Puede incluso referencia el protocolo utilizado por el servidor virtual, como el HTTP. *Aplication port:* El número de puerto. *Protocol:* Ofrece una opción de TCP o UDP, en un menú desplegable, escoger uno de ellos. *Virtual IP Address*: El servidor virtual de la dirección IP flotante. *Virtual IP Network Mark*: La máscara de red para este servidor virtual, en el menú desplegable. *Firewall Mark:* Para entrar en un Mark firewall da valor cuando la agrupación de múltiples protocolos o la creación de un multi-puerto del servidor virtual por separado, pero los protocolos conexos. *Device:* El nombre del dispositivo de red a la que desea la dirección IP flotante se define en la dirección IP virtual de campo obligatorio. Se debe aliar la dirección IP pública con la interfaz ethernet conectado a la red pública. [5]. *Re-entray time*: Un valor que

define el número de segundos antes de que el router activo LVS intente utilizar un verdadero servidor después de que el servidor no real no responda. *Service Timeout*: Un valor que define el número de segundos antes de un verdadero servidor se considere muerto y no está disponible. *Quiesce Server*: Cuando el **servidor Quiesce** botón de opción está seleccionado, en cualquier momento un nuevo servidor real nodo se conecta, las conexiones menores en el cuadro se restablecerá en cero por lo que el router activo LVS solicita rutas como si todos los verdaderos servidores recién se añaden al grupo. Esta opción evita que el nuevo servidor se convierta en atascado con un alto número de conexiones al entrar en el grupo. *Load monitoring tool*: El router LVS puede supervisar la carga en los distintos servidores reales, ya sea mediante el uso de rup o ruptime. If you select rup from the drop-down menu, each real server must run the rstatd service. Si selecciona rup desde el menú desplegable, cada servidor real debe ejecutar el rstatd servicio. Si selecciona ruptime, cada servidor real debe ejecutar el servicio rwhod. *Schedunling* El algoritmo preferido de programación en el menú desplegable en esta solución se debe escoger el WLC (weithing least conection). *Persistence:* Se utiliza en caso de necesitar conexiones persistentes para el servidor virtual del cliente durante las transacciones. Especifica el número de segundos de inactividad permitido antes de la expiración de una conexión. *Persistence Network Mark*: Para limitar la persistencia de subred particular, seleccione la adecuada máscara de red en el menú desplegable. (Ver Anexo 3.6). Al hacer clic en el *Real server subsección*: enlace en la parte superior del panel muestra el *EDIT REAL SERVER* subsección: Se muestra el estado del servidor físico anfitrión de un servicio virtual. Haga clic en el **botón** ADD para añadir un nuevo servidor. Para eliminar un servidor existente, seleccione el botón situado junto a ella y haga clic en el **botón** "DELETE. Haga clic en el **botón** EDIT para cargar el servidor real. El panel de configuración del real server, este grupo se compone de tres campos de entrada. *Name*: Un nombre descriptivo para el servidor real, sugerencia este nombre no es el de host para la maquina, por lo que sea descriptivo y fácilmente identificable. *Address*:. La verdadera dirección IP del servidor. Desde el puerto de escucha ya está especificada para el servidor virtual asociado, no agregue un número de puerto. *Weigth:* Un valor que indica el presente la capacidad de acogida en relación con la de otros anfitriones .El valor puede ser arbitrario, sino pues tratarlo

como una proporción en relación a otros servidores reales.(Ver Anexo 3.7) Haga clic en el vínculo *MONITORINGS SCRIPTS* en la parte superior de la página. La EDICIÓN DE MONITORINGS SCRIPTS subsección permite al administrador especificar un envío / esperar cadena de secuencias para verificar que el servicio de servidor virtual es funcional a cada servidor real. También es el lugar donde el administrador puede especificar scripts personalizados para comprobar los servicios que requieren cambios dinámicos de datos. *Sending Program:* Para obtener avanzados servicios de verificación, puede utilizar este campo, para especificar la ruta a un servicio de control de secuencia de comandos. Esta función es especialmente útil para servicios que requieren cambios dinámicos de datos, tales como HTTPS o SSL. Para utilizar esta función, usted debe escribir un script que devuelve una respuesta textual, sistema que sea ejecutable, y el tipo de camino al mismo en el campo send program, si un programa externo se escribe en el campo Sending Program entonces el campo *Enviar* se ingnora. *Send*: Una cadena para el demonio nanny, para enviar a cada servidor real en este campo. Por defecto el envío se ha completado para HTTP. Se puede alterar este valor en función de sus necesidades. Si deja este campo en blanco, demonio nanny intenta abrir el puerto y asume que el servicio está corriendo si tiene éxito. Sólo está permitido enviar una secuencia de este campo, y que sólo pueden incluir impresora, caracteres ASCII, como los siguientes caracteres de escape: \ n para nueva línea, \ r para el retorno de carro, \ t para la ficha. *Expert:* La respuesta textual que el servidor debe devolver si es que funciona correctamente. Si escribió su propio programa de envío, escriba la respuesta que le dijo a enviar si tuvo éxito. [5].

3.4 El Problema del ARP

Viene dado mediante la configuración de LVS-DR además también ocurre con IP Tunneling. Cuando utilizamos Direct Routing tanto el balanceador como los servidores comparten una dirección IP (VIP) y esto puede traer consigo problemas si los balanceadores y los servidores están en la misma red. El modelo OSI de la ISO consta de 7 capas, las tres primeras son la capa física, la capa de enlace de datos y la capa de red. Cuando un ordenador transmite datos empieza a encapsular en la capa de aplicación, séptima capa. Cuando llega a la capa de red añade la información de red, direcciones IP y direcciones lógicas (origen y destino). Acto seguido añade su dirección física o MAC que es una dirección única que tiene cada

tarjeta de red o NIC y que consta de dos partes una que identifica al fabricante de la tarjeta y la otra parte identifica a la tarjeta. Después en la capa física se traduce toda la información a señales eléctricas y se transmite por el medio. Además de sus direcciones propias IP y MAC se añaden las direcciones del destinatario y si el paquete fuera destinado a una red diferente de la de partida se pondría en el campo de la MAC de destino la MAC de Gateway o del router por defecto y este se iría encargando de enrutar el paquete hasta que llegará al último router o Gateway el cual cambiaría su MAC por la MAC del equipo que tuviera la IP de destino del paquete. Este último router mandaría el paquete por la interface correspondiente y todos los equipos en esa red desencapsularían el paquete hasta la segunda capa y sólo aquel cuya MAC este en esa trama, como destino, tomará el paquete y lo desencapsulará entero para hacer uso de él. Con Direct Routing tenemos que tener en cuenta que los clientes hacen las peticiones al balanceador, pero sin embargo las respuestas las reciben de los servidores. [6]

3.4.1 Importante

Tanto el balanceador de carga como los servidores comparten una IP (VIP), cuando un cliente solicita una conexión con VIP la petición se debe de hacer al balanceador, no a los clientes. Cuando llega una petición de un cliente para el servicio bajo LVS esta llegará desde fuera de la red, con lo cual el router de esa red hará una petición ARP para obtener la MAC de la máquina con la IP VIP. En esa red podría haber varias máquinas con la IP VIP (el balanceador y los servidores comparten dicha IP) con lo cual cualquiera de ellas podría, y de hecho lo hará, responder a la petición. Pero el paquete deberá ir destinado al balanceador no a los servidores. El balanceador registrará en sus tablas a que servidor le manda las peticiones y consecuentemente todas las peticiones de ese cliente irán al mismo servidor, bajo la conexión ya establecida. Si uno de los servidores respondiera a la petición ARP el router tendría en su tabla ARP la dirección física del servidor y todos los paquetes se los enviará directamente al servidor sin utilizar el balanceador. Si en algún momento se cambiará la entrada en la tabla ARP y el router actualizará con la MAC de otra máquina (el balanceador y el resto de servidores tienen una interface o alias con la IP VIP) entonces las peticiones de ese cliente irán a otro servidor en lugar de al servidor que originariamente estaban yendo. Si esto pasa dentro de una misma conexión cuando un servidor empiece a recibir las solicitudes de una

conexión que él no ha iniciado (la realizó el servidor que primero respondió a la petición ARP) la conexión se cerrará y habrá que volver a negociarla. Este problema se presenta con núcleos a partir de la serie 2.2.x y se soluciona haciendo que la interface que tiene la IP VIP no responda a peticiones ARP en los servidores y si en el balanceador de carga. De esta forma nos aseguramos que cuando el router haga una petición ARP para la VIP la única máquina que responda sea el balanceador y de esta forma todos los paquetes para el LVS serán enviados al balanceador y este hará su trabajo. [6]

3.4.2 Las soluciones

Puede abordarse el problema de los ARP de varias maneras, de entre las diferentes soluciones posibles, destacarían los siguientes métodos.
Detener las respuestas desde los servidores reales ante peticiones ARP para resolver la VIP.
- Configurar dispositivos para que no respondan peticiones ARP (NOARP flag).
- Parchear las versiones 2.2.x ó 2.4.x del kernel.
- Ocultar la VIP de los servidores reales para que estas no vean las peticiones ARP.
- Poner una Network Interface Card (NIC) extra en los servidores para soportar la VIP.
- Poner los servidores reales en una red diferente a la de la VIP (Lars' Metod)
- Cebar al router del director con la MAC correcta para la VIP (la de director).
- Hacer que los servidores reales contesten con la MAC del director.
- Detener las peticiones ARP a la VIP hechas a los servidores reales.
- Modificar tabla de routado del router.
- Permitir a los servidores reales aceptar paquetes con una IP destino igual a la VIP, incluso si el servidor real no tiene una interfaz con esa IP. [2]

Se necesitarán 3 máquinas (1 cliente, 1 director, 1 realserver), esto es como mínimo y para hacer pruebas del balanceador pero no es lo más óptimo ya que se debe tener al menos 2 directores para que en caso de que alguno sufra alguna colisión pues el servicio no se caiga y se active el otro. Además para la solución planteada pues se va a utilizar varios servidores reales debido a la cantidad de servicios que se prestan en la universidad.

3.5 Persistencia en LVS

En una situación de balanceo de carga normal, se asume que cada conexión de red es independiente de las demás, de modo que cada conexión puede ser asignada a un servidor diferente si el algoritmo de scheduling así lo cree conveniente. No obstante, en ocasiones hay servicios que requieren que todas las conexiones de un mismo cliente sean asignadas al mismo servidor real, lo cual recibe el nombre de persistencia de sesión. LVS identifica las conexiones de los clientes en base al contenido de las cabeceras de los protocolos IP, TCP y UDP, y utiliza por lo tanto métodos de persistencia basados en IP origen para garantizar la persistencia de los servicios que la requieren. El resultado de aplicar un método de persistencia en LVS, implica que el cliente se conectará al mismo servidor real para diferentes conexiones TCP o UDP, entendiendo como conexión TCP el tiempo que transcurre desde que se establece la conexión TCP a tres bandas hasta que ésta finaliza, ya sea de manera acordada TCP FIN, o de forma abrupta, TCP RESET. Al conjunto de conexiones TCP/UDP realizadas, necesarias para la correcta prestación de un determinado servicio, se le conoce como sesión, y al mecanismo que permite garantizar que todas estas conexiones sean reenviadas a un mismo servidor, como método de persistencia de sesión.

3.5.1 Servicios que requieren aplicar mecanismos para asegurar la persistencia de sesión

Existen tres situaciones en las que es necesario aplicar mecanismos de persistencia para la correcta prestación de un servicio ofrecido LVS:

- Cuando el cliente necesita acceder siempre a al mismo servidor, debido a que sólo éste dispone de la información intercambiada con el cliente durante la prestación de un determinado servicio. Por ejemplo, debido al intercambio de llaves realizado durante una sesión https .
- Si el cliente solicita un servicio que se ofrece a través de dos puertos diferentes, los cuales han de ser abiertos en el mismo servidor real. Por ejemplo, el servicio FTP activo (puertos 20, 21), o el FTP pasivo (puerto 21 y otro puerto alto por encima del 1023), o bien los servicios de un sitio de e-commerce (puertos 80, 443). [2]

Cuando un mismo cliente realiza conexiones pertenecientes a una misma sesión, pero utiliza direcciones IP origen diferente en cada una de las conexiones, debido a que está accediendo a Internet a través de un proxy diferente en cada conexión. Esta situación se la conoce también como el megraproxy problem, la cual es descrita en el apartado del mismo nombre de esta memoria. Una forma de solucionar este problema, es asumiendo que los diferentes servidores proxy por los cuales sale el cliente, pertenecen a la misma red. Entonces, es posible configurar al balanceador para que considere a todas las conexiones, realizadas desde un rango de direcciones de red determinado, como si fueran de un mismo cliente u origen virtual (virtual source). En LVS, se conoce a este tipo de persistencia como persistent granularity.

FTP es un ejemplo muy claro para entender la necesidad del uso de la persistencia. El cliente establece dos conexiones con el servidor, una de ellas es la conexión de control (puerto 21) para el intercambio de la información de control y la otra es la conexión de datos (usualmente el puerto 20) para la transferencia de los datos. En el FTP activo, el cliente informa al servidor del puerto por el que le escuchará, la conexión de datos es inicializada por el servidor desde el puerto 20 y dirigida al puerto que el cliente le informó.

SSL es otro servicio que necesita conectarse siempre al mismo servidor durante toda la sesión debido al uso de las llaves. Cuando se establece una conexión SSL, en el puerto 443 para servidores Web seguros y en el puerto 465 para servidores de correo seguro, una llave para la conexión debe de ser escogida e intercambiada. Las últimas conexiones para el mismo cliente son garantizadas por el servidor durante la vida de la llave SSL. [2]

3.5.2 Mecanismos de persistencia implementados en LVS

LVS no puede interpretar el contenido de los paquetes más allá de la capa de transporte, por este motivo no sabe determinar el inicio y finalización de sesiones. LVS implementa métodos de persistencia basados en IP origen (3.8.2). LVS permite implementar Persistent Client Connection (PCC) o Persistent Port Connection (PPC), en función de la versión del kernel utilizada. De este modo, los kernels anteriores o iguales a la versión 2.2.10 sólo permitían PCC, en cambio los kernels iguales o superiores a la versión 2.2.12 implementan PPC, pudiendo implementar también PCC poniendo un ‘0’ al configurar el número de puerto

mediante la opción –p de ipvsadm (6.7). LVS no es capaz de implementar, por sí sólo, el método de agrupación de puertos descrito en el apartado, pero sí mediante la utilización de marcas de firewall. [2]

3.5.3 Funcionamiento de los mecanismos de persistencia en LVS

La tabla de sesiones utilizada por LVS muestra el estado de las conexiones de los clientes que atraviesan el balanceador. Cada conexión balanceada por LVS se refleja mediante una entrada en esta tabla de sesiones (hash table), constado cada una de ellas de los siguientes campos:

IPVS connection entries

Pro expire state source virtual destination

El primer campo especifica el protocolo TCP o UDP. El segundo indica el tiempo que la regla permanecerá en la tabla, si no se recibe otro paquete que coincida con el resto de campos. Si se recibe un paquete que case con una entrada en la tabla, el campo state puede cambiar de valor. El cambio de estado (state) implica reajustar el valor del contador (timeout), mostrado en el campo expire. La siguiente captura muestra el cambio de estados por los que va pasando una sesión, y el reajuste que del temporizador (expire) en función del cambio:

Pro expire state source virtual destination
TCP 14:56 ESTABLISHED 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80
TCP 14:49 ESTABLISHED 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80
TCP 01:59 FIN_WAIT 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80
TCP 00:05 CLOSE 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80
TCP 00:02 CLOSE 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80
TCP 00:00 CLOSE 192.168.3.157:1044 192.168.3.150:80 192.168.10.101:80

Cuando el balanceador detecta el fin de una conexión TCP no la borra de la tabla, sino que la mantiene durante 10 segundos, pasado este tiempo y si no recibe un nuevo paquete que case con el resto de campos de la tabla (VIP, IP origen y puerto), elimina la entrada. [2]

3.6.4 Particularidades

- Persistencia en ftp

El método más optimizado para ofrecer el servicio ftp, en función de lo descrito hasta ahora, sería recurriendo al agrupado de puertos mediante marcas de firewall, sin embargo, existe otra posibilidad que es la de cargar el módulo ip_masq_ftp (para versiones 2.2.x), o el módulo ip_vs_ftp (en versiones 2.4). Estos módulos permiten ofrecer el servicio ftp en LVS-NAT aplicando únicamente la persistencia a los puertos necesarios para la prestación del servicio ftp (20 y 21 para el modo activo, 21 y un puerto alto para el modo pasivo). Se procederá de la siguiente forma para configurar el servicio ftp en el balanceador de carga (director1).

root@director1:~# modprobe ip_vs_ftp
root@director1:~# lsmod
Module Size Used by Not tainted
ip_vs_ftp 4516 0 (unused)
ip_vs_rr 1476 2 (autoclean)
ip_vs 77272 5 (autoclean) [ip_vs_ftp ip_vs_rr]

Posteriormente, se configurará mediante ipvsadm, el comando que establece el servicio ftp como persistente:
root@director1~#ipvsadm -A -t 192.168.3.150:21 -p 600
root@director1~#ipvsadm -a -t 192.168.3.150:21 -r 192.168.10.101 –m
root@director1~#ipvsadm -a -t 192.168.3.150:21 -r 192.168.10.102 -m
- Persistencia http
Con la primera versión del protocolo http, HTTP 1.0, cuando el cliente solicitaba la descarga de una página, se establecía una conexión TCP para descargar cada uno de los objetos que contenía. No tenía demasiado sentido establecer una conexión TCP para descargar el poco código de la página, cerrar la conexión y volver a establecer conexiones nuevas para descargar el resto de objetos que componen la página (por ejemplo, imágenes). Con HTTP 1.1 las conexiones persistentes son posibles. Los extremos de la comunicación cliente/servidor negocian para ver si la conexión persistente está disponible. El servidor http mantiene la conexión abierta tras haber enviado toda la información solicitada, por un periodo de tiempo de 16 segundos, si pasado este tiempo el navegador del cliente no solicita nuevos

objetos, la conexión se cierra. El cliente puede cerrar la conexión en cualquier momento si así lo desea (por ejemplo cuando ya tenga todos los objetos que conforman la página).

Si se solicita una página web alojada en el clúster desde un navegador tan difundido como pueda ser Microsoft Internet Explorer y posteriormente se visualiza el contenido de la tabla de sesiones del balanceador de carga LVS, se observa lo siguiente:

root@director1:~# ipvsadm -lcn

IPVS connection entries

pro expire state source virtual destination

TCP 14:59 ESTABLISHED 192.168.3.157:1063 192.168.3.150:80 192.168.10.102:80

TCP 14:59 ESTABLISHED 192.168.3.157:1062 192.168.3.150:80 192.168.10.101:80

Se observa que el navegador ha establecido dos conexiones TCP para obtener el contenido de la página web, lo que conlleva a plantearse la siguiente pregunta, ¿Es que Microsoft Internet Explorer no implementa el protocolo HTTP 1.1? Antes de salir corriendo a anunciar esta afirmación a bombo y platillo, sería recomendable continuar leyendo, pues no es eso lo que está sucediendo. Lo que ocurre, es que los navegadores web lanzan múltiples conexiones TCP concurrentes hacia el servidor, en su carrera por ser el navegador más rápido en mostrar las páginas al cliente. Por otra parte, cada una de esas conexiones ha utilizado el protocolo HTTP 1.1 para descargar diferentes objetos, de lo contrario habría establecido tantas conexiones como objetos tuviese la página. [2]

Para finalizar éste apartado, se muestran a continuación algunos ejemplos que permiten configurar algunos servicios de forma persistente utilizando diferentes métodos de reenvío.

Estos ejemplos han sido extraídos de:

<http://www.linuxvirtualserver.org/docs/persistence.html>

- Persistencia http en LVS/NAT

ipvsadm -A -t virtualdomain:www –p

ipvsadm -a -t virtualdomain:www -r 192.168.1.2 –m

ipvsadm -a -t virtualdomain:www -r 192.168.1.3 –m

ipvsadm -a -t virtualdomain:www -r 192.168.1.4 –m

- Persistencia http en LVS/TUN

ipvsadm -A -t virtualdomain:www –p

ipvsadm -a -t virtualdomain:www -r 192.168.1.2 –l
ipvsadm -a -t virtualdomain:www -r 192.168.1.3 –l
ipvsadm -a -t virtualdomain:www -r 192.168.1.4 –l
- Persistencia ftp en LVS/DR
ipvsadm -A -t virtualdomain:ftp -p 540
ipvsadm -a -t virtualdomain:ftp -r 192.168.1.2 –g
ipvsadm -a -t virtualdomain:ftp -r 192.168.1.3 –g
ipvsadm -a -t virtualdomain:ftp -r 192.168.1.4 –g
- Persistent Client Connection (PCC) en LVS/DR
ipvsadm -A -t virtualdomain:0 –p
ipvsadm -a -t virtualdomain:0 -r 192.168.1.2 –g
ipvsadm -a -t virtualdomain:0 -r 192.168.1.3 –g
ipvsadm -a -t virtualdomain:0 -r 192.168.1.4 –g 72
- Persistent granularity (Megaproxy Problem)
ipvsadm -A -t virtualdomain:www -p -M 255.255.255.0
ipvsadm -a -t virtualdomain:www -r 192.168.1.2
ipvsadm -a -t virtualdomain:www -r 192.168.1.3
ipvsadm -a -t virtualdomain:www -r 192.168.1.4

3.6 Hardware que se requiere

Cliente: Cualquier máquina, cualquier sistema operativo, con un cliente telnet y / o un cliente http. (netscape, lynx, IE). **Director (2):** máquina con sistema operativo Linux para este caso Red Hat parcheado con ipvs. Si estás empezando desde cero, use un núcleo 2.6.10 y asi se evita tener que parchear el kernel. Que trae consigo en los paquetes de ipvsadm y piranha-gui, esta máquina no necesita ser fuerte, si se debe garantizar un buena tarjeta de Red ya que el tráfico de peticiones por esta PC va a ser de una amplia magnitud y con 1gb de RAM para que el entorno web del piranha y el trabajo del LVS no se vea afectado por ningún problema de bajo rendimiento. **Realserver (s):** Estas son las máquinas que ofrecen el servicio de los intereses). En estos entornos de producción puede tener cualquier sistema operativo. El

director de la posibilidad de reenviar los paquetes de 3 métodos para la realservers puede ser LVS-NAT - LVS-DR - LVS-Tun, pero en este caso pues Direct Routing es la solución.

3.7 Monitorización de los servidores

Los balanceadores de carga se configuran para reenviar las conexiones de unos determinados servicios a los servidores reales encargados de ofrecerlos. Para que un balanceador de carga ofrezca alta disponibilidad de los servidores reales, es necesario que actualice su configuración cuando se produce un fallo en alguno de los nodos de la granja de servidores reales. Para poder cambiar dinámicamente esta configuración, se hace necesaria la utilización de algún sistema de seguimiento del estado de los servidores y de sus aplicaciones. Estos sistemas monitorizan los servicios ofrecidos por cada servidor real y realizan las modificaciones necesarias en la configuración del balanceador, cuando detectan alguna anomalía. En función del tipo de monitorización que realizan, estos sistemas se pueden clasificar como sistemas in-band o como sistemas out-band.

3.7.1 Sistemas in-band y Sistemas out-band

Los sistemas in-band se caracterizan por utilizar el propio tráfico cliente-servidor para controlar el estado del servidor real. Cuando se detecta un problema, es posible iniciar una serie de controles más exhaustivos para determinar si el fallo se ha producido por la caída del servidor que estaba ofreciendo el servicio. Si es así, se reconfigura oportunamente al balanceador para que deje de reenviar peticiones a ese servidor. Los sistemas out-band generan su propio tráfico para monitorizar la salud de los servidores reales, que pueden ir desde la simple comprobación de la conectividad, a nivel de capa de enlace de datos, a la comprobación del correcto funcionamiento de los servicios de red ofrecidos y las aplicaciones que los prestan.[2]

3.7.2 Métodos para monitorizar la salud del servidor (Health Checks)

Chequeo básico

Dentro de esta categoría entrarían aquellos chequeos que recurren a pruebas de conectividad de las capas inferiores de la pila OSI: Peticiones ARP en la capa de enlace de datos,

peticiones hecho ICMP (ping) en la capa de red o peticiones a un puerto TCP/UDP determinado, en la capa de transporte. Estas pruebas son tanto más efectivas a medida que se sube de nivel en la capa OSI, si bien, son siempre un método incapaz de determinar con las garantías suficientes si un servicio está funcionando correctamente. Por otra parte, este método no permite realizar ningún tipo de control sobre los contenidos. Por ejemplo, en el caso de que un servidor web sirva una página de indicación de error (404 URL no encontrada), el sistema de monitorización interpretará que el servidor muestra las páginas correctamente, ya que el sistema no efectúa ningún tipo de comprobación de los contenidos. En otro tipo de servicios, la simple contestación a una petición, aunque ésta sea con un código de error, bastaría para que el sistema los considerase activos. [2]

Chequeo específico de las aplicaciones

Este tipo de chequeos pretenden determinar el correcto funcionamiento de una aplicación del servidor. La técnica más utilizada en este tipo de chequeo, consiste en realizar una petición del servicio y analizar la información obtenida en respuesta a la misma. De esta manera, es fácil determinar si la aplicación está respondiendo como se espera o no de ella. Por ejemplo, para chequear un servidor web, puede enviarse una petición HTTP y observar que el resultado de la respuesta obtenida concuerda con el esperado. Un chequeo a un servidor FTP, podría consistir en conectarse a éste e identificarse con el correspondiente LOGIN y PASS. Si se obtiene acceso al servicio, el sistema interpretará que la aplicación funciona correctamente. En ocasiones, la correcta prestación de un servicio requiere del correcto funcionamiento de los servicios que se prestan por otros puertos. Por ejemplo, en el caso de un sitio web de e-comerce, si el servicio https (puerto 443 SSL) no funciona correctamente, tampoco tiene sentido ofrecer el servicio http (puerto 80), ya que normalmente dependen uno del otro. En estos casos, puede recurrirse al agrupado de puertos, de manera que si uno de los puertos del grupo falla en alguno de los servidores reales, se considere como fallidos a todos los servicios del grupo. [2]

Chequeo de contenidos

El método de chequeo de contenidos se basa en la misma filosofía que la descrita anteriormente en el sistema de chequeo de aplicaciones. En este caso, el sistema también puede buscar determinados patrones o palabras clave, en la respuesta enviada por el servidor. Un método usado también para este fin, es el de efectuar checksums de la información recibida. En ocasiones, el chequeo de una sola aplicación en el servidor, no es suficiente para determinar su correcto funcionamiento cuando ésta depende de otras. Por ejemplo, puede estar funcionando el servicio web y estar fallando las consultas que éste realiza a la base de datos. Para evitar este tipo de situaciones, podría realizarse un chequeo adicional de los contenidos. Un chequeo de contenidos de una base de datos, podría efectuarse llevando un poco más allá el concepto de chequeo de aplicaciones. Así, una forma de resolver el problema descrito en el ejemplo anterior, sería realizar una petición de comprobación al servidor http, que debería retornar el contenido de una determinada base de datos. De no ser así, se interpretaría que las consultas a la base de datos en cuestión están fallando. [2]

Chequeo basado en agentes

Otra posibilidad, es la de utilizar sondas o agentes para monitorizar la salud de las aplicaciones que corren en los servidores. Estas sondas envían la información recogida al controlador que corre en el balanceador. Mediante la API que proporcionan estos sistemas, es posible programar “alarmas” que disparen los mecanismos para reconfigurar el balanceador en caso necesario. Muchos de estos sistemas de chequeo son compatibles con estándares, como por ejemplo, SNMP. La elección de uno de estos métodos de monitorización de servidores o la combinación de varios de ellos, estará en función de las necesidades concretas para cada caso particular. No obstante, lo ideal sería optar por la decisión más equilibrada entre necesidades, recursos hardware disponibles, servicio que se quiere prestar y coste económico. [2]

Conclusiones

En el capítulo se pudo apreciar la forma en la cual trabaja el ipvsadm los comandos y los parámetros que se necesitan para su uso efectivo, además una guía paso por paso de cómo

trabajar con el Piranha-Gui sobre todo muy específicamente en cada consola de trabajo, que significa cada detalle que se puede ver en el entorno web, para hacer un fácil uso de sus opciones y configuración. En la instalación y utilización de este software (LVS) pues se puedo apreciar un problema con las peticiones ARP y también se dieron varias soluciones de cómo terminar con esa problemática y de manera general pues también se explica de cómo monitorear los servidores.

CONCLUSIONES

Para la consecución de los objetivos iniciales del LVS, se ha visto la necesidad de incrementar los conocimientos en sistemas operativos así como en la de ganar soltura en labores de configuración. No sólo fue necesario saber implementar soluciones que resolvieran los objetivos planteados al iniciar el proyecto, utilizando para ello software libre, sino que también hubo que documentar debidamente el proceso de instalación y configuración. Se realizo un estudio de este software en el mundo en Cuba y en la UCI.

Finalmente se cumplieron los objetivos trazados en el trabajo, se selecciono y configuro la técnica y el algoritmo para ser utilizado por el balanceador y se documento paso por paso como utilizar las herramientas que lo componen.

Por otra parte, el modelo en el cual se cimenta el software libre, que ha permitido desarrollar todas la aplicaciones necesarias para construir el clúster LVS, demuestra su madurez para implementar soluciones que respondan a problemas reales. Como muestra de ello pueden observar la gran gama de sitios web que actualmente utilizan esta solución. (Ver Anexo 1.2).

RECOMENDACIONES

A modo de recomendaciones pues decir que sería muy bueno hacerle las pruebas de carga a esta herramienta usando un script y a través de una aplicación usando el método experto y así podemos saber cuál es la capacidad y el alcance que tiene, cuantas peticiones pudiera soportar el balanceador para tener una muestra de cuanta calidad puede tener para su trabajo.

REFERENCIA BIBLIOGRAFICA

1. Efraim Wainerman "SERVIDORES VIRTUALES IMPLEMENTADOS EN EL SISTEMA OPERATIVO LINUX" 2002 www.buanzo.com.ar/files

2. Marcos Martínez Jiménez "Arquitecturas de Clustering de Alta Disponibilidad y Escalabilidad (LinuxVirtualServer)"V1.0.1 Marzo 2003 http://www.scribd.com/doc/2925766/acadelvsmemoria

3. "Clustering de Alta Disponibilidadbajo GNU/Linux" Vicente José Aguilar Roselló Septiembre 2001 www.bisente.com/documentos/clustering/memoria.html

4. http://www.linuxvirtualserver.org/

5. Red Hat Cluster Suite: Overview 1.10. Linux Virtual Server Administration GUI http://www.redhat.com/docs/manuals/csgfs/browse/rh-cms-desc-ov-en/s1-lvsmgmttools-overview.html

6. José Angel de Bustos Pérez "Utilización y Administración avanzadas de sistemas GNU/Linux y aplicaciones Software Libre para estudiantes universitarios Clustering y Alta Disponibilidad en GNU/Linux" 15-May-2007 http://es.tldp.org/Manuales-LuCAS/doc-curso-salamanca-clustering/

BIBLIOGRAFIA

1. http://www.i-excom.com/balanceadores-de-carga/#caracteristicas

2. http://linux-vserver.org/

3. http://riseuplabs.org

4. http://www.redhat.com/support/wpapers/piranha

5. http://supersparrow.sourceforge.net

6. http://www.nl.ultramonkey.org

7. http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/

8. Ana Molina. [Citado: 25/05/2007.] http://www.rcci.net/globalizacion/2004/fg435.htm.

9. Proyecto GNU y Fundación de Software Libre: “Filosofía del Proyecto GNU”. [Citado el: 1/11/2006.] http://www.gnu.org/philosophy/philosophy.es.html.

10. Stallman, Richard: “Porqué "Software Libre" es mejor que software de "Código Fuente Abierto". [Citado el: 24/11/2006.] http://www.gnu.org/philosophy/free-software-for-freedom.es.html

11. Raymond, Eric S.: “La Catedral y el Bazar”. [Citado el: 04/11/2006.] http://www.sindominio.net/biblioweb/telematica/catedral.html.

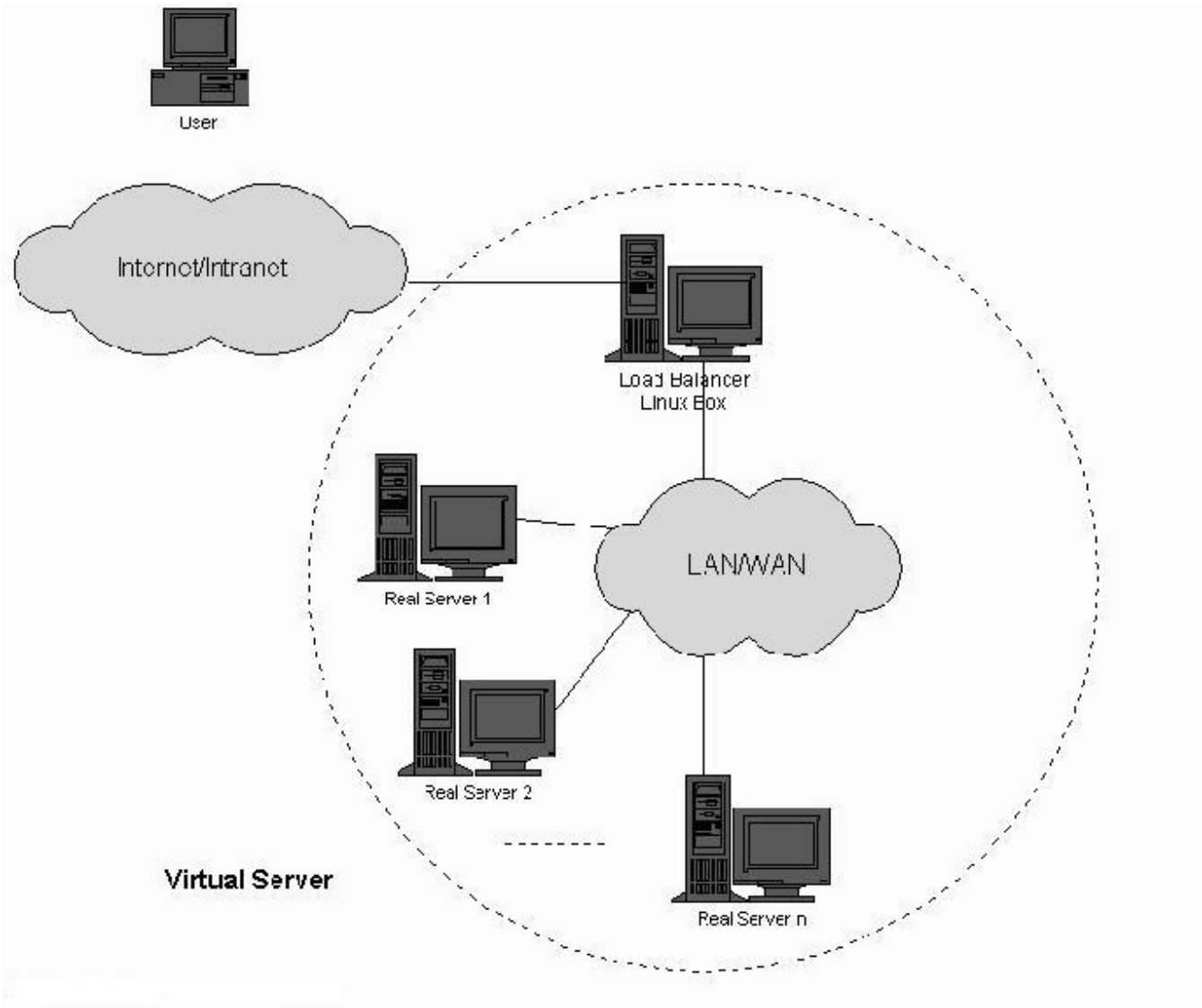
12. Proyecto GNU y Fundación de Software Libre: “La Definición de Software Libre”. [Citado el: 17/11/2006.] http://www.gnu.org/philosophy/free-sw.es.html

.

ANEXOS

Anexo 1.1

LVS esquema general.

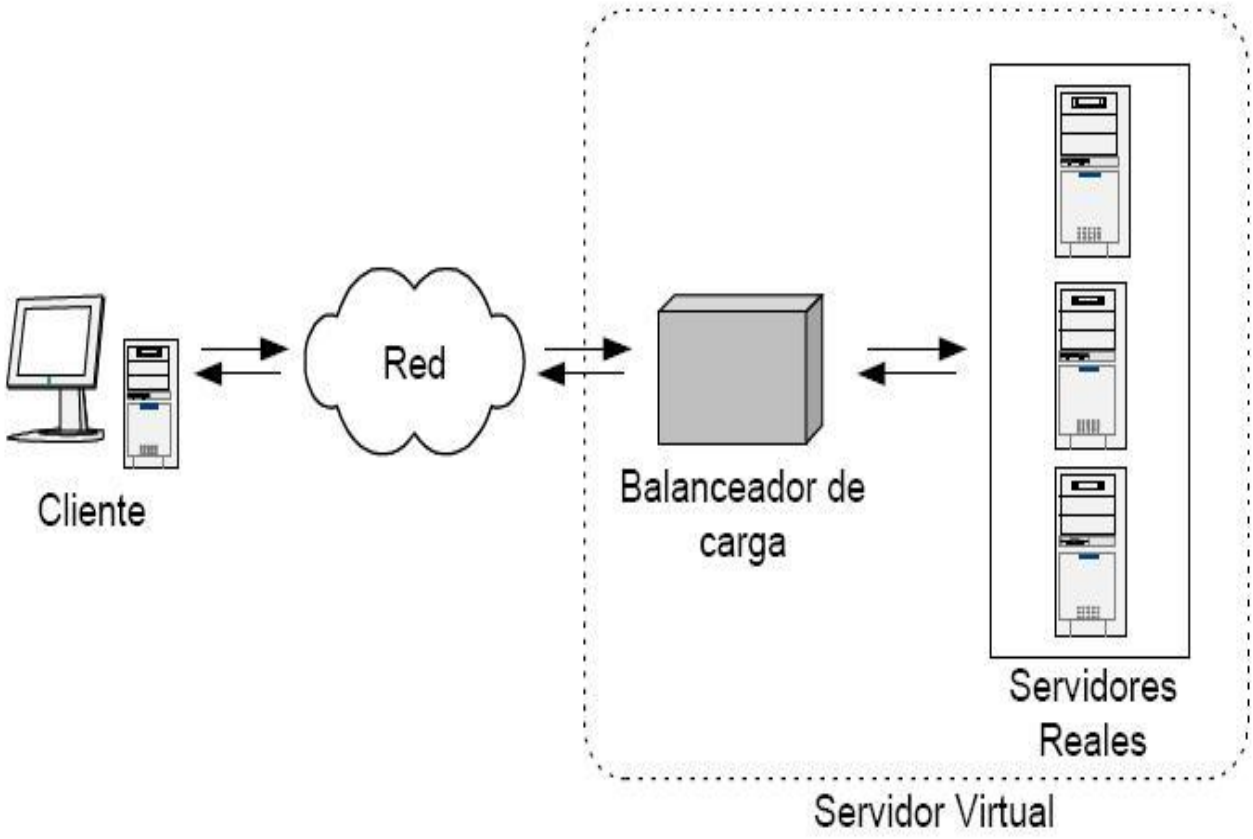


Anexo 1.2

El siguiente cuadro está extraídos del sitio de LVS (http://www.linuxvirtualserver.org), en el se recoge un listado de diferentes sitios web y proyectos que han sido implementados con el servidor virtual linux (LVS).

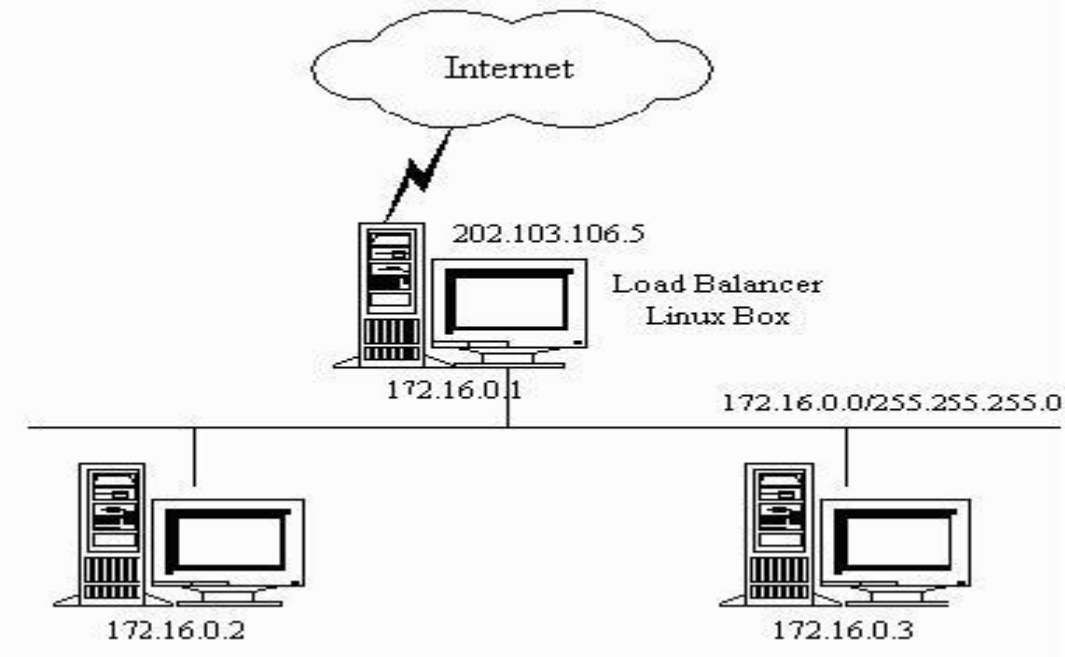
Site	Description	LVS Notes
linux.com	The Linux Portal	LVS *web* cluster
sourceforge.net	The largest open source development site	LVS (load balancing HTTP, HTTPS, FTP, SSH, CVS)
themes.org	The largest interface/themes site for the X Window System	LVS *web* cluster
wwwcache.ja.net	UK National JANET *Web* Cache Service	40 Squid cache servers in three LVS clusters, ~900GB content shipped/day. for more information
www.zope.org	The leading Open Source *web* application server	*web* cluster, for more information
masks.org	Online mask museum	A LVS cluster of four Pentium and Pentium II class computers. More information
TuxFamily.org	Free hosting for free *software* like sourceforge and savannah	using LVS/heartbeat (2 load balancers failover and 8 real servers) for all the hosting services
www.care2.com	Largest *web* site community of people who care about the environment	6+ *web* servers and a single load balancer by Jeremy Brand. more information
valinux.com	VA Linux Systems, Inc.	LVS *web* cluster
www.real.com	Real Networks, Inc.	LVS media cluster (>20 nodes)
www.netwalk.com	A full service Internet Access and Presence Provider	LVS *web* hosting cluster
www.greenheck.com	Leading manufacturer of air movement and control equipment	LVS *web* cluster (5 node). Cliff Baeseman provides two pictures (1, 2) of the cluster
map.net	An ICP that allows you to explore the Internet in the same way you explore a map	LVS *web* cluster (6 nodes) by Brian Edmonds
anandtech.com	a big *hardware* review site	two primary-backup load balancers and eleven *web* servers, more information
www.tiscover.com	The 3rd largest Portal Site in Austria with more than 500,000 visits per day	LVS *web* cluster
www.cycosmos.com	An virtual community *web*site in United Kingdom and Germany	LVS *web* cluster

Anexo 1.3



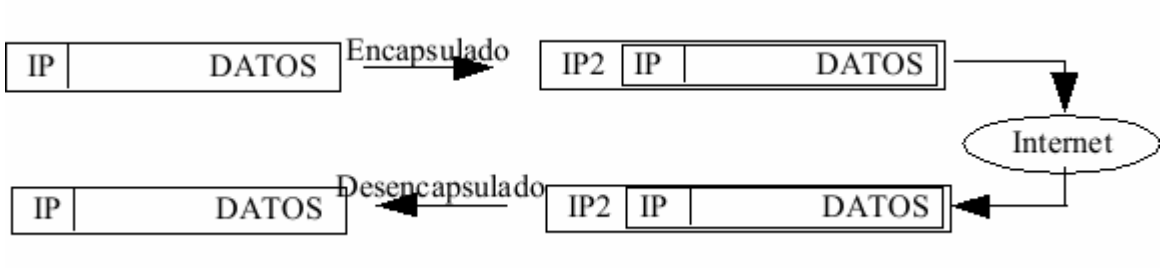
Ubicación del balanceador de carga.

Anexo 2.1



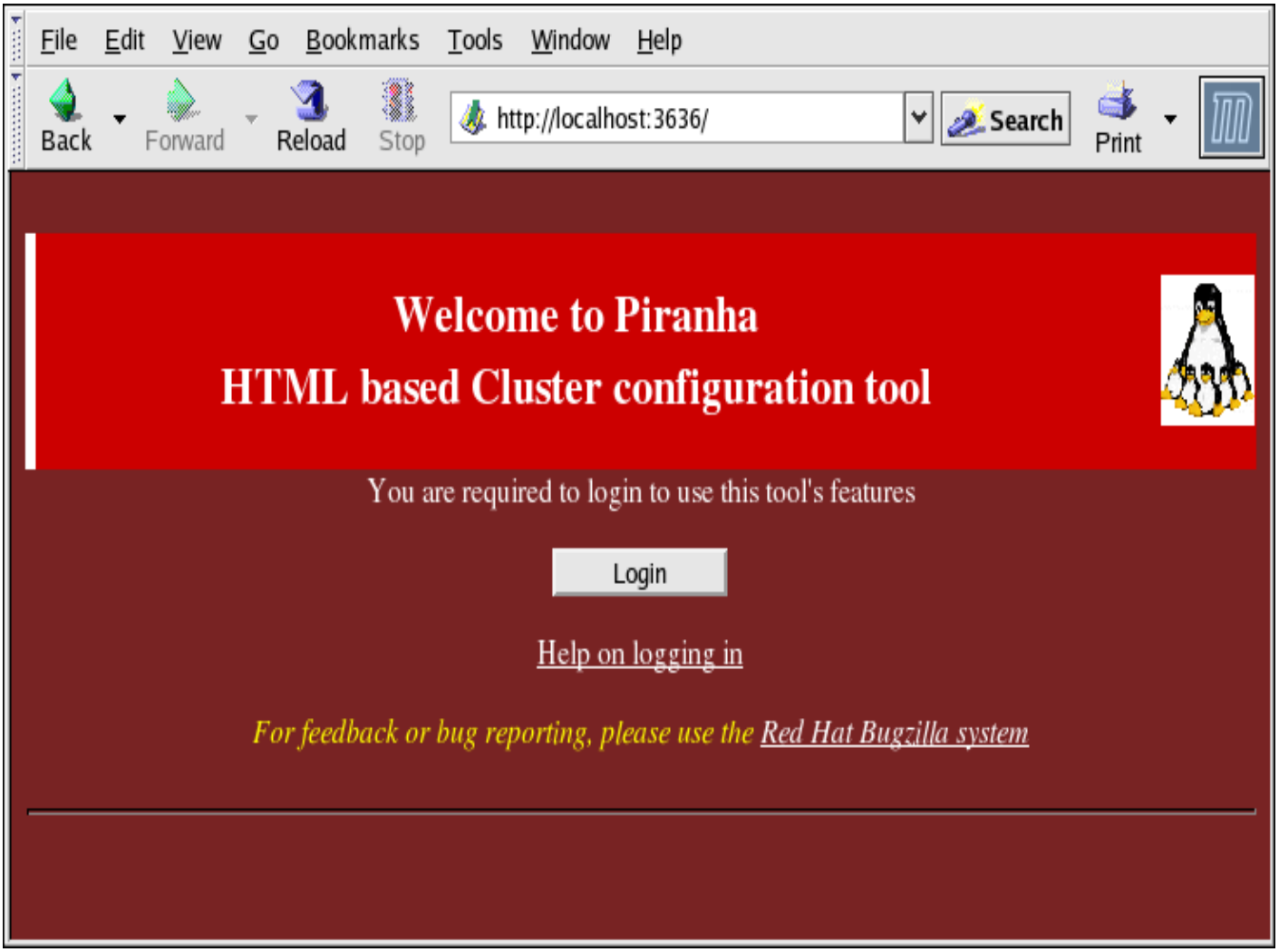
LVS-NAT, esquema físico

Anexo 2.2

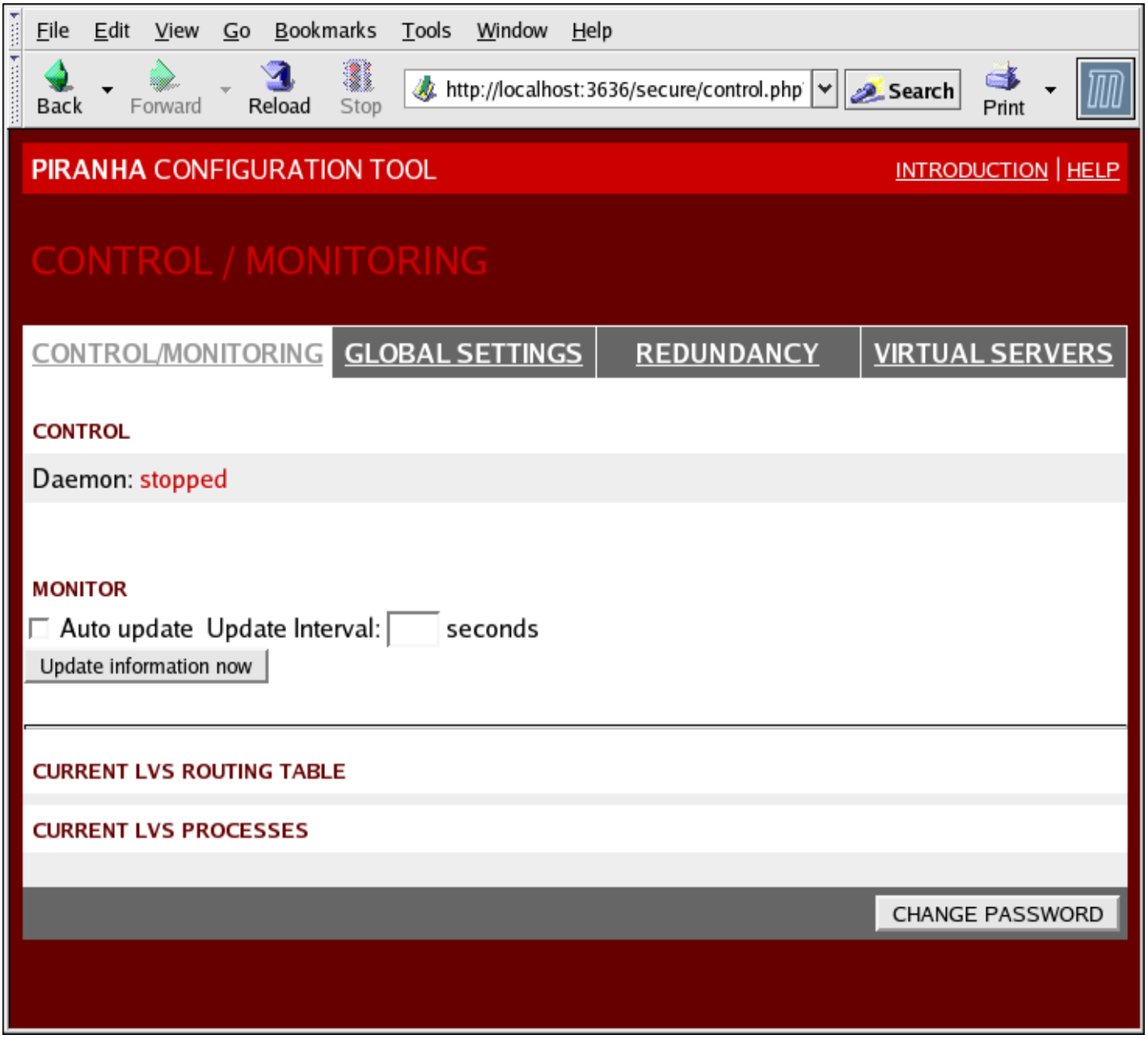


LVS-IP Tunneling- encapsulado y desencapsulado.

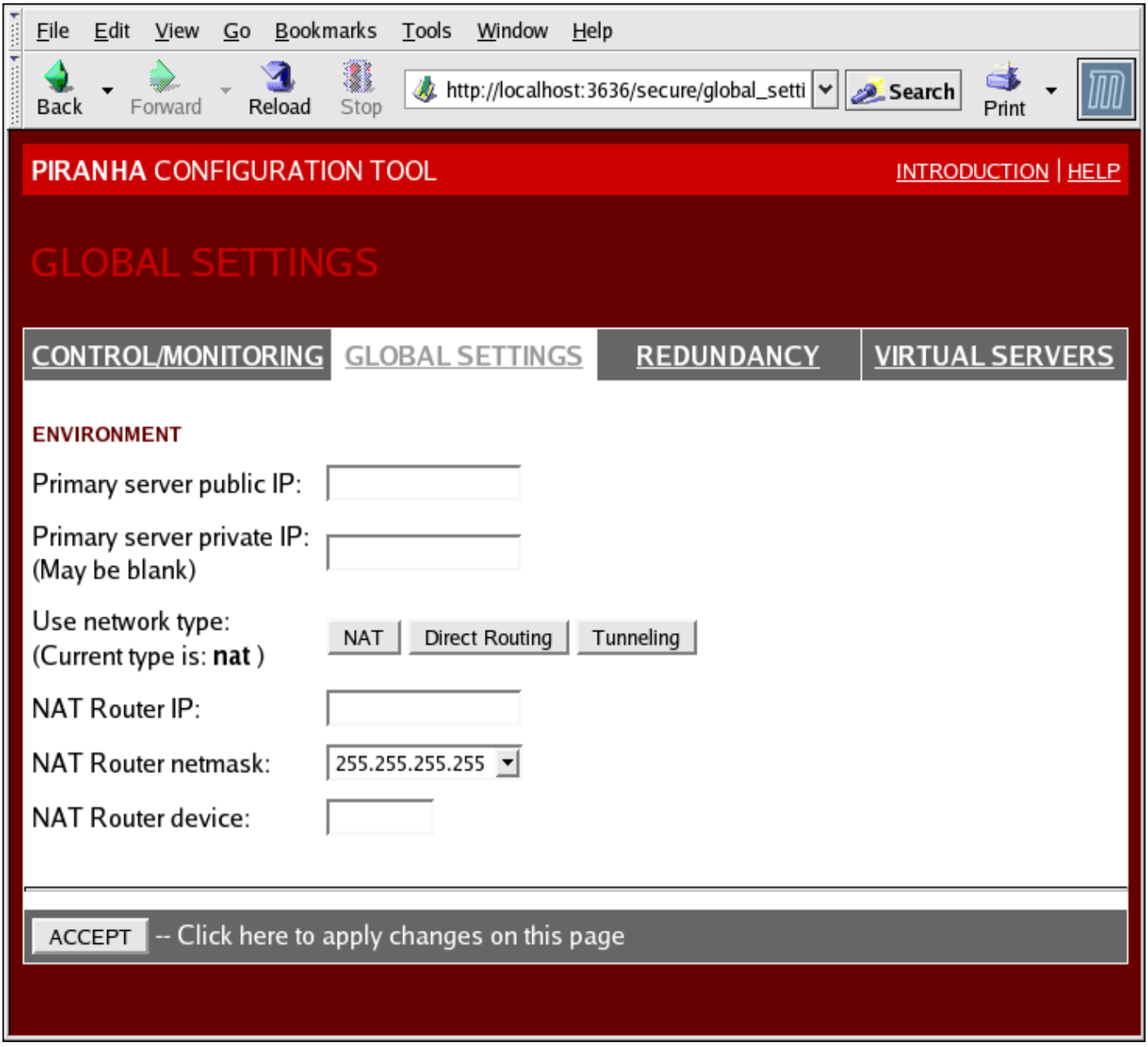
Anexo 3.1


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/
Search
Print
Welcome to Piranha
HTML based Cluster configuration tool
You are required to login to use this tool's features
Login
Help on logging in
For feedback or bug reporting, please use the Red Hat Bugzilla system

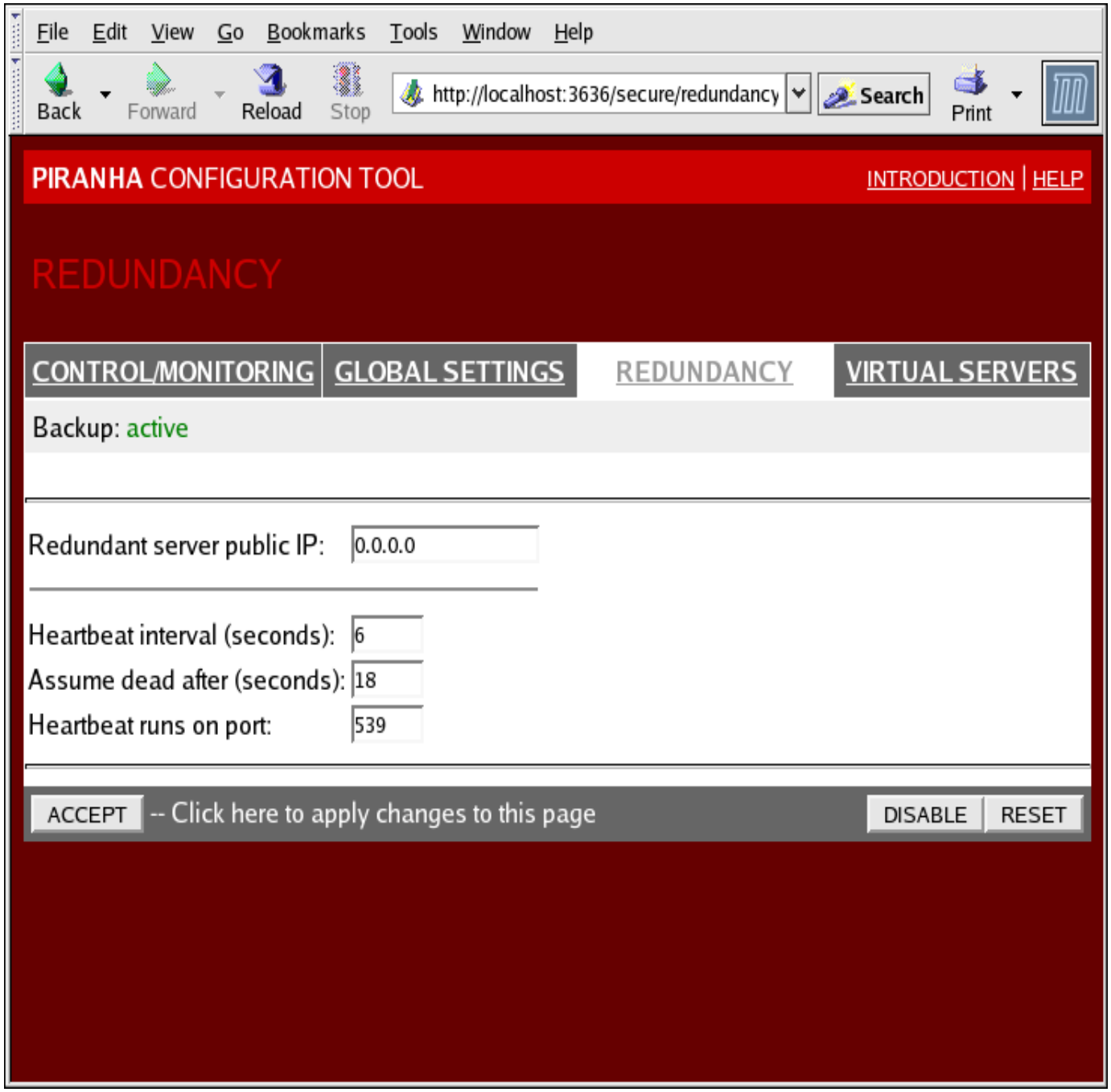
Anexo 3.2


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/control.php
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
CONTROL / MONITORING
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
CONTROL
Daemon: stopped
MONITOR
Auto update Update Interval: seconds
Update information now
CURRENT LVS ROUTING TABLE
CURRENT LVS PROCESSES
CHANGE PASSWORD

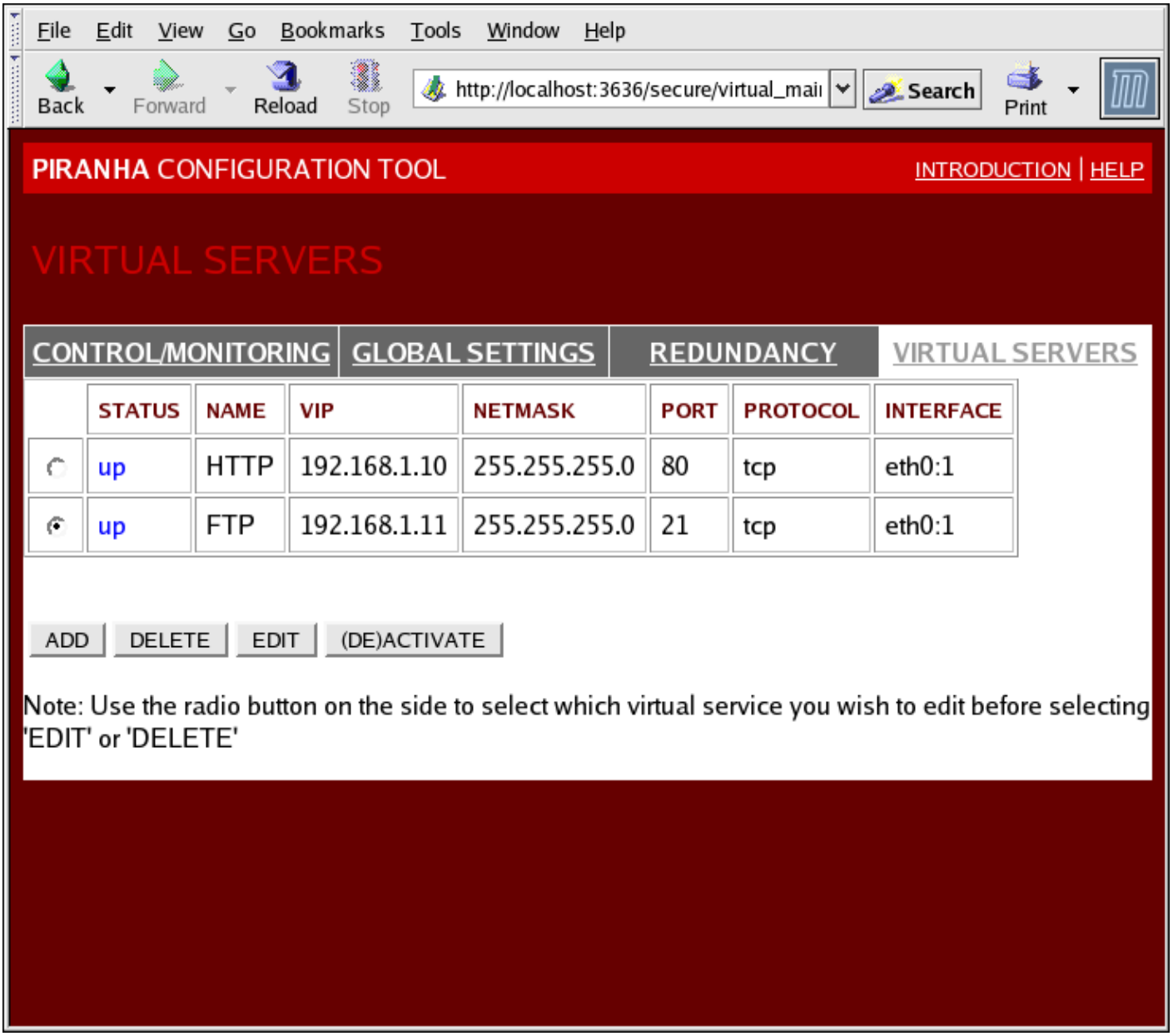
Anexo 3.3


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/global_setti
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
GLOBAL SETTINGS
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
ENVIRONMENT
Primary server public IP:
Primary server private IP:
(May be blank)
Use network type:
(Current type is: nat)
NAT
Direct Routing
Tunneling
NAT Router IP:
NAT Router netmask:
255.255.255.255
NAT Router device:
ACCEPT
-- Click here to apply changes on this page

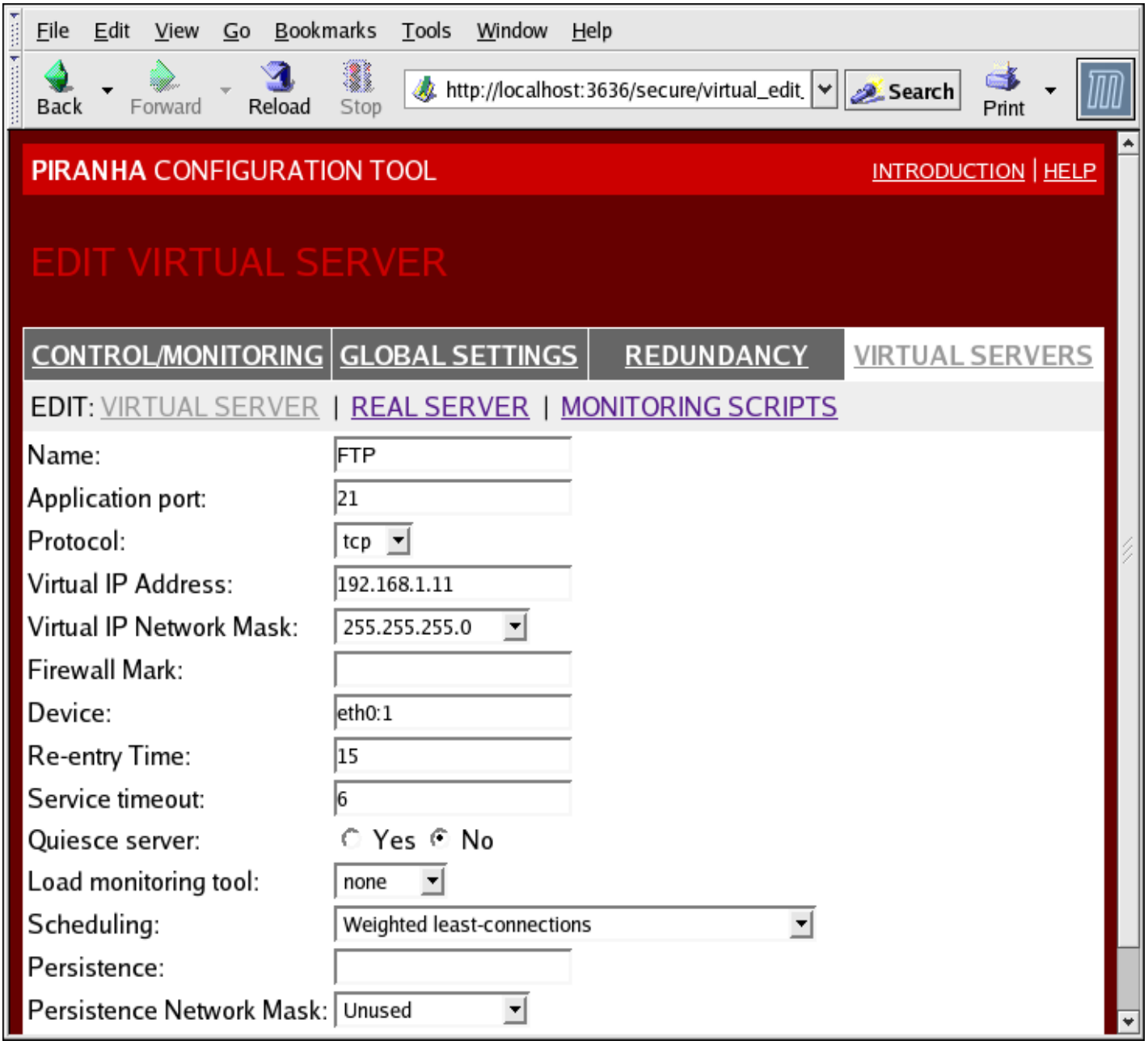
Anexo 3.4


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/redundancy
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
REDUNDANCY
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
Backup: active
Redundant server public IP:
0.0.0.0
Heartbeat interval (seconds):
6
Assume dead after (seconds):
18
Heartbeat runs on port:
539
ACCEPT
-- Click here to apply changes to this page
DISABLE
RESET

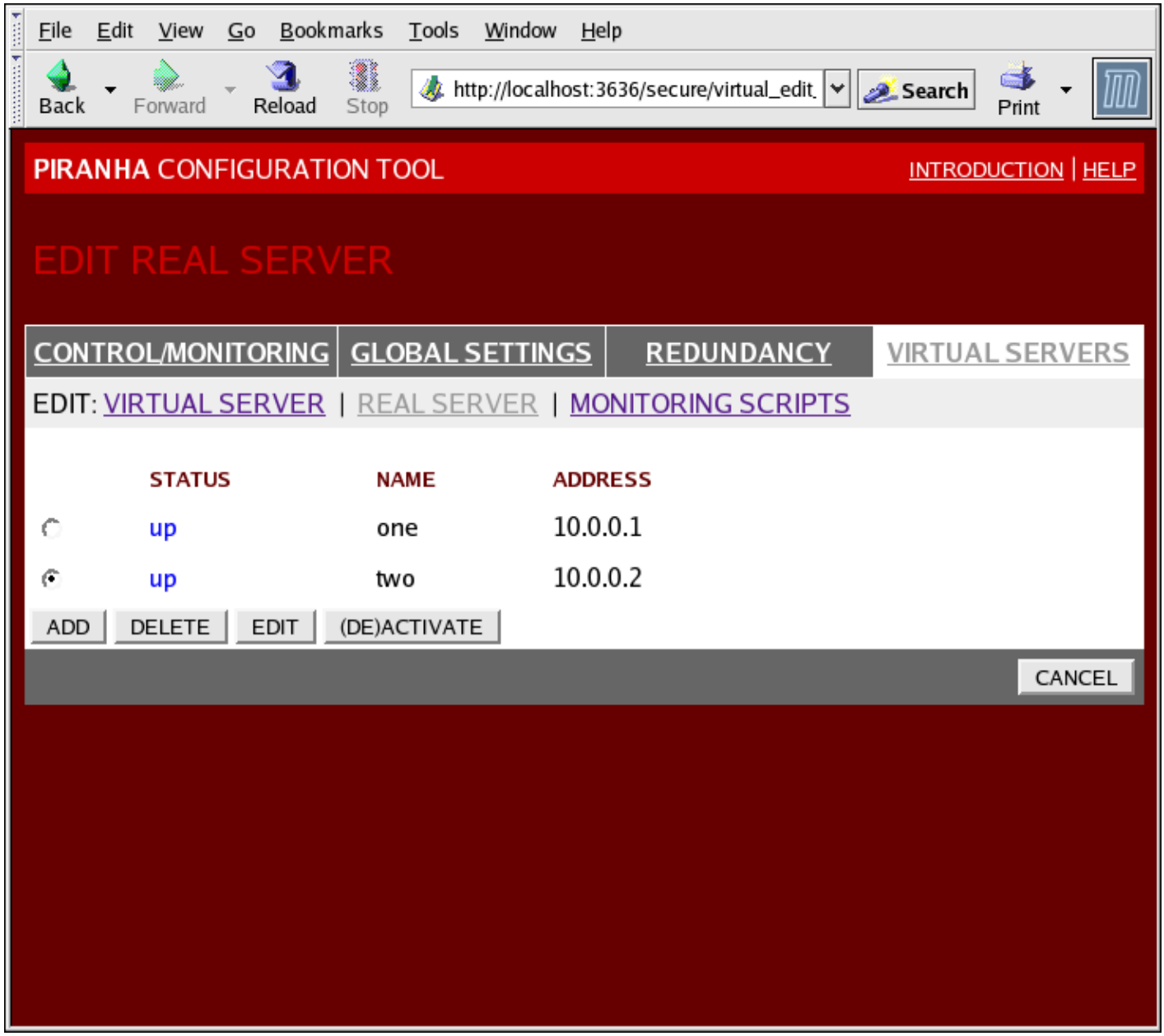
Anexo 3.5


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_mai
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
VIRTUAL SERVERS
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
STATUS NAME VIP NETMASK PORT PROTOCOL INTERFACE
up HTTP 192.168.1.10 255.255.255.0 80 tcp eth0:1
up FTP 192.168.1.11 255.255.255.0 21 tcp eth0:1
ADD DELETE EDIT (DE)ACTIVATE
Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

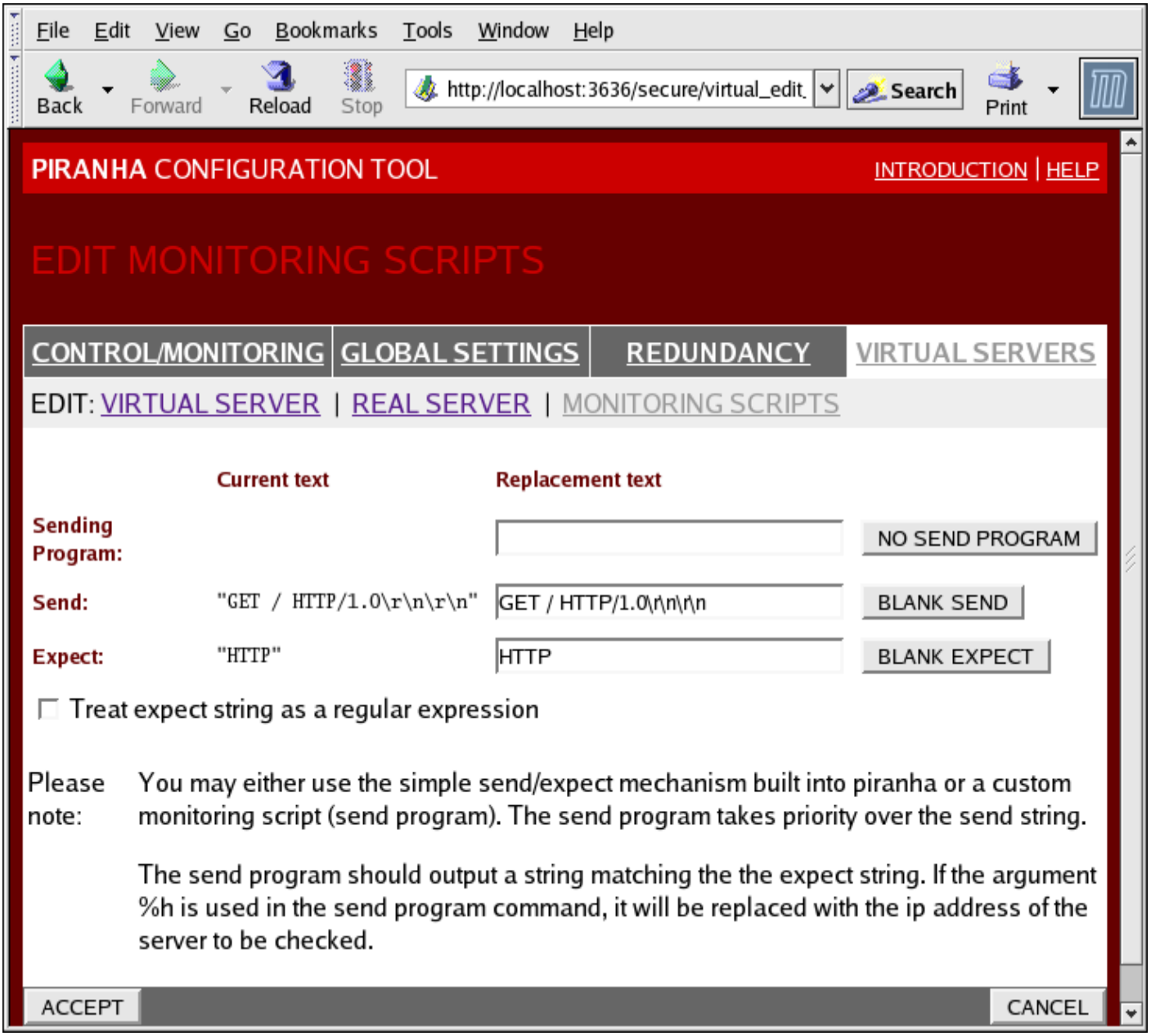
Anexo 3.6


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_edit
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
EDIT VIRTUAL SERVER
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
Name: FTP
Application port: 21
Protocol: tcp
Virtual IP Address: 192.168.1.11
Virtual IP Network Mask: 255.255.255.0
Firewall Mark:
Device: eth0:1
Re-entry Time: 15
Service timeout: 6
Quiesce server: Yes No
Load monitoring tool: none
Scheduling: Weighted least-connections
Persistence:
Persistence Network Mask: Unused

Anexo 3.7


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_edit
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
EDIT REAL SERVER
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
STATUS
NAME
ADDRESS
up
one
10.0.0.1
up
two
10.0.0.2
ADD
DELETE
EDIT
(DE)ACTIVATE
CANCEL

Anexo 3.8


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_edit.
Search
Print
PIRANHA CONFIGURATION TOOL
INTRODUCTION | HELP
EDIT MONITORING SCRIPTS
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUAL SERVERS
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
Current text
Replacement text
Sending Program:
NO SEND PROGRAM
Send:
"GET / HTTP/1.0\r\n\r\n"
GET / HTTP/1.0\r\n\r\n
BLANK SEND
Expect:
"HTTP"
HTTP
BLANK EXPECT
Treat expect string as a regular expression
Please note:
You may either use the simple send/expect mechanism built into piranha or a custom monitoring script (send program). The send program takes priority over the send string.
The send program should output a string matching the the expect string. If the argument %h is used in the send program command, it will be replaced with the ip address of the server to be checked.
ACCEPT
CANCEL

GLOSARIO DE TERMINOS

Escalabilidad: La escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. En general, también se podría definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

Disponibilidad: clúster de alta disponibilidad es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí. Podemos dividirlo en dos clases:

Balanceo de carga: balance o balanceo de carga es un concepto usado en informática que se refiere a la técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos. Está íntimamente ligado a los sistemas de multiprocesamiento, o que hacen uso de más de una unidad de procesamiento para realizar labores útiles. El balance de carga se mantiene gracias a un algoritmo que divide de la manera más equitativa posible el trabajo, para evitar los así denominados cuellos de botella que es el objetivo del multiprocesamiento.

Procesamiento paralelo: El procesamiento en paralelo es una forma eficaz de de procesamiento de información que favorece la explotación de los sucesos concurrentes en el proceso de computación. Concurrencia implica paralelismo simultaneidad. Se basa en la utilización de equipos con múltiples procesadores que pueden ejecutar instrucciones concurrentemente y comparten el acceso a los datos locales o pudiera verse como utilizar varios equipos independientes que colaboran entre sí para cumplir un objetivo.

Clúster: Un clúster de balanceo de carga o de cómputo adaptativo está compuesto por uno o más ordenadores (llamados nodos) que actúan como frontend del clúster, y que se ocupan de repartir las peticiones de servicio que reciba el clúster, a otros ordenadores del clúster que forman el back-end de éste.

Overhead: Término coloquial de origen inglés, que se utiliza para denominar el proyector de transparencias o Tiempo de proceso necesario para que se ejecuten los comandos antes de que un dispositivo esté listo para dar acceso.

ARP: El Address Resolution Protocol (protocolo de resolución de direcciones) para la resolución de direcciones en informática, responsable de encontrar la dirección hardware que corresponde a una determinada dirección IP.

NAT: NAT (Network Address Translation - Traducción de Dirección de Red) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles.

Direct-Routing: Significa ruta directa y es una de las técnicas que utilice LVS para balancear la carga.

Ip Tunneling: Encapsulamiento Ip consiste en encapsular un datagrama IP dentro de otro y redirigirlo a otra máquina. La máquina receptora debe desencapsular el paquete.

Algoritmos Sheduling: Son los algoritmos de programación implementados para que el balanceador de carga LVS dirija el trabajo de manera que le sea configurado, existen 10 algoritmos de este tipo.

Piranha-GUI: Piranha-gui es una herramienta de configuración basada en la Web para la carga el Piranha Servidor del balanceador. Está escrito en lenguaje PHP, y es usado en el sistema operativo Red Hat.

Persistencia: En una situación de balanceo de carga normal, se asume que cada conexión de red es independiente de las demás, de modo que cada conexión puede ser asignada a un servidor diferente si el algoritmo de scheduling así lo cree conveniente. No obstante, en ocasiones hay servicios que requieren que todas las conexiones de un mismo cliente sean asignadas al mismo servidor real, lo cual recibe el nombre de persistencia de sesión.

SSL: (Secure Socket Layer) para permitir confidencialidad y autenticación en Internet. SSL es una capa por debajo de HTTP y tal como lo indica su nombre esta a nivel de socket por lo que permite ser usado no tan solo para proteger documentos de hipertexto sino también servicios como FTP, SMTP, TELNET entre otros. La idea que persigue SSL es encriptar la comunicación entre servidor y cliente mediante el uso de llaves y algoritmos de encriptación.

API: Una API representa una interfaz de comunicación entre componentes software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla

SNMP: El Protocolo Simple de Administración de Red o SNMP es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Es parte de la familia de protocolos TCP/IP. SNMP permite a los administradores supervisar el desempeño de la red, buscar y resolver sus problemas, y planear su crecimiento.

Monitorización: Es el proceso por medio del cual, nos aseguramos que nuestro proceder está encaminado adecuada y eficazmente hacia un resultado final, evitando las posibles desviaciones que pudieran presentarse. La monitorización puede detectar las posibles interferencias que pudieran presentarse en el curso de alguna acción y puede dar lugar a corregir el procedimiento antes de llegar a un resultado final.

ICMP: El Protocolo de Mensajes de Control de Internet o ICMP (por sus siglas de Internet Control Message Protocol) es el subprotocolo de control y notificación de errores del Protocolo de Internet (IP). Como tal, se usa para enviar mensajes de error, indicando por ejemplo que un servicio determinado no está disponible o que un router o host no puede ser localizado ICMP difiere del propósito de TCP y UDP ya que generalmente no se utiliza directamente por las aplicaciones de usuario en la red. La única excepción es la herramienta ping y traceroute, que envían mensajes de petición Echo ICMP (y recibe mensajes de respuesta Echo) para

determinar si un host está disponible, el tiempo que le toma a los paquetes en ir y regresar a ese host y cantidad de hosts por los que pasa.

LVS-NAT: Es el término que se utiliza para especificar que se está usando Linux virtual server a través de la vía NAT

LVS-DR: Es el término que se utiliza para especificar que se está usando Linux virtual server a través de la Direct-Routing

LVS-Tun: Es el término que se utiliza para especificar que se está usando Linux virtual server a través de la técnica IpTunneling.

Cliente: Cualquier máquina, cualquier sistema operativo, con un cliente telnet y / o un cliente http.

Director: Máquina con un sistema operativo que tenga instalado los paquetes necesarios para utilizar el balanceador de carga, debe contar con una buena tarjeta de red.

Real server: son las máquinas que ofrecen el servicio de los intereses). En estos entornos de producción puede tener cualquier sistema operativo. Pueden brindar servicios web, servicios proxy o base datos.

MAC: la dirección MAC (Media Access Control address o dirección de control de acceso al medio) es un identificador de 48 bits que corresponde de forma única a una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC. Las direcciones MAC son únicas a nivel mundial, puesto que son escritas directamente, en forma binaria, en el hardware en su momento de fabricación.

Servicio QoS: QoS o Calidad de Servicio (Quality of Service, en inglés) son las tecnologías que garantizan la transmisión de cierta cantidad de datos en un tiempo dado (throughput). Calidad de servicio es la capacidad de dar un buen servicio.

IPVS: un parche para el kernel del equipo que vaya a hacer de balanceador

Modelo OSI: modelo de referencia de Interconexión de Sistemas Abiertos (OSI, Open System Interconnection) lanzado en 1984 fue el modelo de red descriptivo creado por ISO; esto es, un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Gateway: (puerta de enlace) es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino. Una puerta de enlace es normalmente un equipo informático configurado para hacer posible a las máquinas de una red local (LAN) conectadas a él de un acceso hacia una red exterior, generalmente realizando para ello operaciones de traducción de direcciones IP (NAT: Network Address Translation). Esta capacidad de traducción de direcciones permite aplicar una técnica llamada IP Masquerading (enmascaramiento de IP), usada muy a menudo para dar acceso a Internet a los equipos de una red de área local compartiendo una única conexión a Internet, y por tanto, una única dirección IP externa.

Spam: Se llama spam, correo basura o sms basura a los mensajes no solicitados, habitualmente de tipo publicitario, enviados en grandes cantidades (incluso masivas) que perjudican de alguna o varias maneras al receptor. Aunque se puede hacer por distintas vías, la más utilizada entre el público en general es la basada en el correo electrónico. También se llama spam a los virus sueltos en la red y páginas filtradas (casino, sorteos, premios, viajes y pornografía), se activa mediante el ingreso a páginas de comunidades o grupos o acceder a links en diversas páginas.

Kernel: se refiere al núcleo (en inglés: kernel) de un sistema operativo.

Loopback: El dispositivo de red loopback es un interfaz de red virtual que siempre representa al propio dispositivo independientemente de la dirección IP que se le haya asignado. El valor en IPv4 es 127.0.0.1 y ::1 para el caso de IPv6.Se utiliza en tareas de diagnóstico de

conectividad y validez del protocolo de comunicación, así como para indicar que el destino del puntero o URL es el mismo host.

Datagrama: Un datagrama es un fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia el equipo terminal de datos receptor, de manera independiente a los fragmentos restantes.